NANYANG TECHNOLOGICAL UNIVERSITY

GOSSIP ALGORITHM IN WIRELESS SENSOR NETWORK

LU FENG

SCHOOL OF COMPUTER ENGINEERING

2008

GOSSIP ALGORITHM IN WIRELESS SENSOR NETWORK

LU FENG

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Master of Engineering

2008

Acknowledgments

I am deeply indebted to my thesis adviser Prof. Chia Liang-Tien, Clement for his encourage, help, and support in so many ways; especially his absolute trust when I encountered the initial difficulties in research.

I would like to thank Prof. Chan Syin and Prof. Lee Keok Kee for their invaluable discussions and suggestions during weekly project meetings. I am also glad to have the opportunity to work with a number of excellent final year students from School of Computer Engineering.

I have to express my appreciation to Guan Wei, Han Shuguo, Wu Min and many others for their concerns, kindness and help during my stay in Center for Multimedia and Network Technology (CeMNet). Finally, my sincere gratitude goes to my wife and family members for their love, care, and understanding.

Contents

Abstract

The wireless sensor networks (WSNs) of the near future are envisioned to consist of hundreds to thousands of inexpensive wireless nodes, each with some computational power and sensing capability, operating in an unattended mode. They are intended for a broad range of environmental sensing applications from vehicle tracking to habitat monitoring. The applications, networking principles and protocols for these systems are still being actively researched.

The problem of distributed consistency maintenance was introduced in the early 1980s. The system is assumed to include n piece of duplicated data $m_i (i = 1, 2, \ldots, n)$ located at distributed networked nodes. Each piece of information might be updated and the objective is for content consensus to be reached at all nodes. However, a common drawback of many existing gossip solutions is the low energy efficiency when passing redundant information over the network. Thus gossip algorithm needs to re-engineered in order to be applicable for energy constraint network such as wireless ad hoc and sensor networks. In this regard, NBgossip algorithm is proposed in this thesis, which is based on network coding and neighborhood gossip. In NBgossip, nodes do not simply forward messages they received, instead, the linear combinations of the messages are sent out. In addition, every node exchanges messages with its neighboring nodes only.

Sensor networks are event-based systems that differ from traditional networks in several ways: sensor networks have severe energy constraints, redundant low-rate data, and many-to-one flows. Therefore, data aggregation has been put forward as a particular useful paradigm for wireless routing and network monitoring in sensor networks. The idea is to combine the data coming from different sources enroute - eliminating redundancy, minimizing the number of transmissions and thus save energy. This paradigm shifts the focus from traditional address-centric approaches (TCP/IP/Internet) to a more data-centric approach. Consider a network of n nodes, in which each node u_i holds a piece of measurement data x_i, and all nodes are required to compute the average of n sensor measurements. This problem is known as averaging problem and is identified to be an important step in data aggregation. It is of particular interest to many applications, such as measurement noise removal, sensor calibration, and distributed data fusion. In this thesis, we shall propose a novel algorithm, which formulates the data aggregation process as a closest point search in a n-dimensional cube and models the communication process as a random gossip. We demonstrate that the true average can be computed

in the optimal $O(\log n)$ and our approach outperforms existing approachers in terms of number of radio transmissions.

Once deployed in the physical world, even factory-calibrated sensors are prone to numerous faults that corrupt sensor data. For example, salinity sensors are particularly susceptible to bio-fouling (the growth of biological material on the sensor), which gradually degrades sensor performance and corrupts critical data. In a large sensor network, it is cumbersome to detect and fix these errors manually and in a timely fashion. In certain cases, the sensor nodes may be deployed in a physically inaccessible environment. Therefore, automatical sensor data calibration is required. Once the sensor measurements are collected after running the gossip algorithms proposed earlier, we make use of the statistical dependencies of spatial-correlated sensor nodes and suggest a loopy belief propagation approach to calibrate the collected sensor data.

List of Figures

List of Tables

Chapter 1

Introduction

Wireless sensor networks (WSNs) are one of the first real-world examples of pervasive computing, the notion that small, smart, and cheap sensing and computing devices will eventually permeate the environment. Sensor networks combine distributed sensing, computation, and wireless communication. Sensor networks are touted as being as disruptive [1] and enabling a technology such as the Internet, with broad applications such as tracking bushfires and microclimates, monitoring zebras, conducting military surveillance, letting businesses monitor and control their work spaces, monitoring public exposure to contaminants, managing land use, and supporting safer structures.

Wireless sensors have been available for decades. Until recently, however, the development and adopting of the technology has been hampered due to the cost of the sensors. The first driving factor for WSNs is the prolonged exponential growth in the underlying semiconductor technology. The number of transistors on a cost-effective chip doubles every year or two, following Moore's law [2]. Consequently, a given computing capacity becomes exponentially smaller physically and cheaper for at least another 10 to 20 years before it eventually reaches some fundamental technical and economic limit [3]. The same semiconductor manufacturing techniques driving this miniaturization can also be used to build extremely small radios, as well as mechanical structures that sense fields and forces in the physical world. The second driving factor is the miniaturization of energy capacity. Over the past 20 years, a AA nickel alkaline battery's capacity has risen from 0.4 to 1.2 Ah with fast recharging [4]. Moreover, the power consumption of a circuit is very strongly related to its performance capacity. A circuit can almost always be designed to require less energy to complete a task if given more time to complete it. The final driving factor is system-on-a-chip (SoC) integration technology. Microsensors, onboard processing, and wireless interface can be integrated at a very small scale and at relative low power. In 5 to 10 years, "complete systems with computing, storage, communication, sensing, and energy storage could be as small as cubic millimeter leading to very cheap, small form factor sensor devices" [5].

There lies the potential for sensor networks. Sensor networks link the information technologies, the wireless technologies that have revolutionized communications, and the sensor technologies that have revolutionized medicine and industrial process control. Sensor devices can be deeply embedded and densely deployed to enable up-close monitoring of a wide range of physical phenomena. This enables spatial and temporally dense environmental monitoring. The amplitude of most physical signals attenuates sharply with distance. Deeply embedded sensor networks can therefore "reveal phenomena that were previously unobservable" [6]. In the 1990s, the Internet transformed the way in which individuals and organizations interact with each other. Sensor networks promise to transform the way in which we interact with our physical world.

1.1 WSN Network Architecture

Unlike conventional wireless networks, a WSN network has to support large number of sensors in a local area with short range and low average bit-rate communication (fewer than 1-100kbps). The network design has to address the requirement of servicing dense sensor distribution, emphasizing recovery of environmental information. In WSN, as a rule, we seek to exploit the short-distance separation between nodes to provide multi-hop communication through the power advantages gained. Since for short hops, transceiver power consumption for reception is nearly the same as that of transmission, the protocol should be designed so radios are off as much of the time as possible. That is, a device's medium access control (MAC) address in a network should include some variant of time division multiple access.

A time-division protocol requires that the radios exchange short message periodically to maintain local synchronism. It is not necessary for all nodes to have the same global clock, but the local variations from link to link should be small to minimize the guard times between slots and enable cooperative signal processing functions, including fusion and beam-forming. The message can combine network performance information, maintenance of synchronization, and reservation requests for bandwidth by longer packets. The abundant bandwidth resulting from the spatial reuse of frequencies and local processing ensures relatively few conflicts in these requests, so simple mechanisms can be used. At least one low-power protocol suite embodying these principles has been developed, including boot-up, MAC, energy-aware routing, and interaction with mobile units [7]. Its development indicates the feasibility of achieving distributed low-power operation in a flat multi-hop network.

Also it is clear that for a wide range of applications, some way has to be found to conveniently link sensor networks to the Internet. Inevitably, some layering of the protocols (and devices) is needed to make use of these standard interfaces. For example,

the WSN next-generation node architecture design addresses the constraints on robust operation, dense and deep distribution, inter-operability with conventional networks and databases, operating power, scalability, and cost. WSN gateways provide support for the WSN network and access between conventional network physical layers and their protocols and between the WSN physical layer and its low-power protocols. WSN system design exploits the reduced link range available through multi-hopping to provide advantages that system architect can choose from the following set: reduced operating power, improved bit rate, improved bit error rate, improved communication privacy, simplified protocols, and reduced cost. These benefits are not obtained simultaneously but need to be extracted individually, depending on design emphasis.

In network design today, architects also have to address: How can Internet protocols, including TCP and IPv6, be employed within sensor networks? While it is undesirable to develop new protocols or perform protocol conversion at gateways, several factors demand custom solutions. First, IPv6 is not truely self-assembling; while address can be obtained from a server, this particular protocol presupposes attachment at lower levels already. Second, present-day Internet protocols take little account of the unreliability of physical channels or the need to conserve energy, focusing instead on supporting a wide range of traffic. Embedded systems can achieve far higher efficiencies by exploiting the traffic's limited nature. Another question have to address is: Where should the processing and storage take place? Communication costs a great deal compared to processing; therefore energy constraints dictate doing as much processing at the source as possible. Moreover, reducing the amount of data to transmit simplifies network design significantly, permitting scalability to thousands of nodes per Internet gateway. A typical WSN architecture is shown in Figure 1.1. Inside this network, sensor nodes communicate to each other, while aggregated information and computation intensive tasks are routed to gateway nodes. The gateway nodes will report data back to control station via the backbone network and process the computation heavy task for sake of the whole network.

1.2 WSN Node Architecture

A sensor node, also known as a mote, is a node in a WSN which is capable of performing some processing, gathering sensory information and communication with other connected nodes in the network. The typical architecture of sensor node is shown in Figure 1.2. The main components of a sensor node as seen are processing unit, transceiver, external memory, power source and sensor board (see Figure 1.2).

Microcontroller performs tasks, processes data and controls the functionality of other components in the sensor node. Other alternatives that can be used as a controller are: General purpose desktop microprocessor, Digital signal processors, Field Programmable

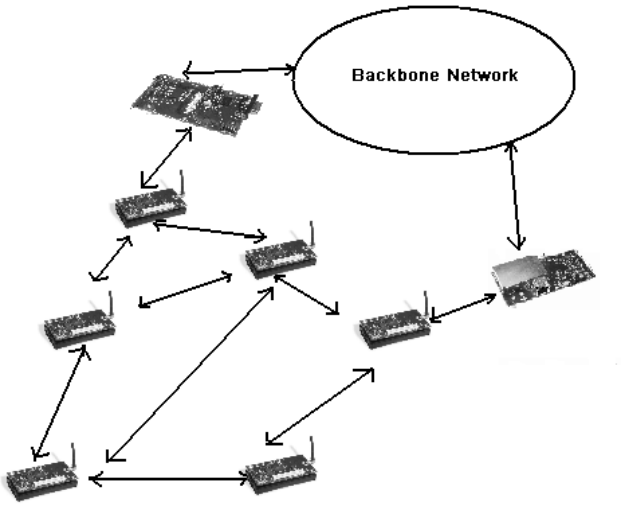

Figure 1.1: Wireless sensor network architecture.

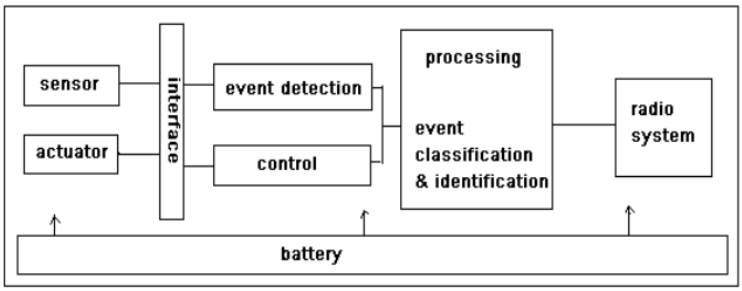

Figure 1.2: Wireless sensor node architecture.

Gate Array (FPGA) and Application-specific integrated circuit. Microcontrollers are most suitable choice for sensor node. Because of their flexibility to connect to other devices, programmable, power consumption is less, as these devices can go to sleep state and part of the controller can be active. Some examples of microanatomies for sensor node are the ATMEGA series and ARM series.

Sensor nodes make use of ISM band [7] which gives free radio, huge spectrum allocation and global availability. The various choice of wireless transmission media are radio frequency (RF), optical communication (Laser), and etc. Laser requires less energy,

but needs line of sight for communication and also sensitive to atmospheric conditions. Infrared like laser, needs no antenna but is limited in its broadcasting capacity. Radio Frequency (RF) based communication is the most relevant that fits to most of the WSN applications. WSN's use the communication frequencies between about 433 MHz and 2.4 GHz. The functionality of both transmitter and receiver are combined into a single device as transceivers are used in sensor nodes. Transceivers lack unique identifier. The operational states are transmit, receive, idle and sleep. Current generation radios have a built-in state machine that perform this operation automatically. Radios operating in idle mode results in power consumption, almost equal to power consumption in receive mode. Thus it is better to completely shutdown the radios rather than in the idle mode when it is not transmitting or receiving.

From an energy perspective, the most relevant types of memory are on-chip memory of a microcontroller and FLASH memory (ROM), while off-chip RAM is rarely used. Flash memories are used due to its cost and storage capacity. Memory requirements are very much application dependent. Two categories of memory based on the purpose of storage are defined: user memory used for storing application related or personal data, and program memory used for programming the device. Power consumption in the sensor node is mainly for sensing, communication and data processing. More energy is required for data communication in sensor node. Energy expenditure is less for sensing and data processing. Power is stored in batteries or capacitors. Batteries are the main source of power supply for sensor nodes.

Sensors are hardware devices that produce measurable response to a change in a physical condition like temperature and pressure. The continual analog signal sensed by the sensors is digitized by Analog-to-Digital (ADC) converter and sent to controllers for further processing. Characteristics and requirements of sensor node should be small size, consume extremely low energy, operate in high volumetric densities, be autonomous and operate unattended, and be adaptive to the environment. Sensors can be classified into the following three categories.

- Passive, omnidirectional sensors: passive sensors sense the data without actually manipulating the environment by active probing. Which means, energy is needed only to amplify their analog signal.

- Passive, narrow-beam sensors: these sensors are passive but they have well-defined notion of direction of measurement. Typical example is "camera".

- Active sensors: these group of sensors active probe the environment, for example, a sonar or radar sensor or some type of seismic sensor, which generate shock waves by small explosions. They are more commonly referred as "actuators".

For most of cases, passive, omnidirectional sensors are considered in WSN. Each sensor node has a certain area of coverage for which it can reliably and accurately report the particular phenomena that it is observing. Several sources of power consumption in sensors are: signal sampling and conversion of physical signals to electrical ones, signal conditioning, and analog to digital conversion.

There are two types of sensor nodes used in sensor network. One is the normal sensor node deployed to sense the phenomena and the other is gateway node or cluster head node that is more powerful and interfaces to external world.

1.3 Application Design Principles

Current applications for WSN are mostly research prototypes or are custom made for a specific purpose. They are insufficiently mature to deploy widely in real world scenarios. At the current stage of research, there is still limited generic off-the-shelf smart sensor nodes, and there is not yet a generic application, which fulfills vastly diverse objectives. There is no typical WSN structure and architecture, and the basic goals of a WSN largely depend on the application. The difficulty of sensor network research is constructing an open system that allows for the variety of real-world phenomena. Hardware constraints, such as the relation between capacity and power consumption, as well as other resources, must be taken into account. To deal with the complexity of sensor network research, Estrin et al. [6] propose two design principles. First, a data centric paradigm where the focus is managing sensor data instead of managing nodes. Secondly, an application-specific approach, where designing the physical and social characteristics of an application environment reduces the domain of discourse of the research. Much sensor network research tends to focus on specific hardware with efficient, ingenious communication control algorithms and system control architectures that address the specific resource constraints. A generic architecture does not emerge yet. The following aspects of the characteristics of applications are identified in order to help in designing WSN applications. The aspects discussed are based on [8], extended with various data processing factors. Applying the right design approaches and technologies to the WSN applications is critical.

1.3.1 Deployment, Mobility, and Infrastructure

Sensor nodes may be installed at specific locations or be placed randomly. After the initial deployment, sensors may be added or replaced, which affects node location, density, and the overall topology. Programs for each sensor node may be pre-installed manually or automatically at runtime [9].

Sensor nodes may be attached to or carried by mobile entities. Mobility may be either an incidental side effects, or it may be a desired property of the system (e.g. to move nodes to interesting physical locations), in which case mobility may be either active (i.e. automotive) or passive (e.g. attached to a moving object not under control of sensor node). Mobility may apply to all nodes within a network or only to a subset of nodes. The degree of mobility may also vary from occasional movement with long periods of immobility in between, to constant travel. Mobility has a large impact on the expected degree of network dynamics and hence influences the design of networking protocols and distributed algorithms. The actual speed of movement may also have an impact.

The various communication modalities can be used in different ways to construct a communication network. Two common forms are so-called infrastructure based networks and ad hoc networks. In infrastructure based networks, sensor nodes can only communication directly with base stations. The number of base stations depends on the communication range and the area covered by the sensor nodes. In ad hoc networks, nodes can communicate directly with each other without an infrastructure. Nodes may act as routers, forwarding messages over multiple hops on behalf of others. Note that cost should not be forgotten. State of the art technology allows a Bluetooth radio system to cost less than 10 dollars. The cost of a sensor node should be much less than 1 dollar in order for the sensor network to be feasible.

1.3.2 Network Topology, Density, Connectivity, and Lifetime

One important property of a WSN is its diameter, that is, the maximum number of hops between any two nodes in the network. In its simplest form, a WSN forms a single hop network, with every sensor node being able to communication directly with every other node. An infrastructure based network with a single base station forms a star network with a diameter of two. A multi-hop network may form an arbitrary graph, but often an overlay network with a simple structure is constructed such as a tree or a set of connected stars. The topology affects many network characteristics such as latency, robustness, and capacity. The complexity of data routing and processing also depends on the topology.

Given the large number of nodes and their potential placement in locations that are not easily accessible, it is essential that the network is able to self-organize; manual configuration is not feasible. Moreover, nodes may fail (either from lack of energy or from physical destruction), and new nodes may join the network. Therefore, the network must be able to reconfigure itself periodically. Furthermore, while individual nodes may become disconnected from the rest of the network, a high degree of connectivity must be maintained, which directly relates to the density of the network.

The effective range of the sensors defines the coverage area of a sensor node. The density of the nodes indicates the degree of coverage of an area of interest by sensor

nodes. The network size affects reliability, accuracy, and data processing algorithms. The density can range from a few nodes to hundreds in a region. The density μ is calculated as in [10]:

$$\mu(R) = (N\pi R^2)/A$$

Where N is the scattered sensor nodes in region A, and R is the radio transmission range. Generally, $\mu(R)$ gives the number of nodes within the transmission radius of each node in region A.

The communication ranges and density of sensor nodes define the connectivity of a network. If there is always a network connection between any two nodes (e.g. via multi-hop routing), the network is said to be connected. Connectivity is intermittent if the network may be partitioned occasionally. If nodes are isolated most of the time and enter the communication range of other nodes only occasionally, we say that communication is sporadic. Depending on application, the required lifetime of a sensor network may range from a few hours to several years. The necessary lifetime has a high impact on the required degree of energy efficiency and robustness of the nodes, thereby requiring the minimization of energy expenditure.

1.3.3 Data Processing

The ultimate goal of a WSN is to detect specific events of interest or collect data in a particular sensor field. Either case suggests WSN being a data centric paradigm.

Data aggregation is the task of data summarization while data are traveling through the sensor network. An excessive number of sensor nodes can easily congest the network, flooding it with information. The prevailing solution to this problem is to aggregate or fuse data within the WSN then transmit an aggregate of the data to the controller. There are three major ways of performing data aggregation: First, diffusion algorithms assume that data are transmitted from one node to the next, thus propagating through the network to the destination. Along the way data may be aggregated, mostly with simple aggregation functions and assuming homogeneous data. Second, streaming queries are based on SQL extensions for continuous querying. Here data are considered to be transient while the query is persistent. Third, event graphs work on streams of events and compose simple events into composite events based on an event algebra. One might think of breaking a database query into simple data transactions for an analogy. The event algebras from reactive middleware have been extended with temporal constraints for event correlation and transmission probabilities for WSNs. Events are consumed according to event consumption modes.

The different approaches above influence when and where the aggregation of data in WSNs should be performed. This involves how to deal with time in distributed and unreliable environments, with state, asynchrony and unstable communication, redundancy

and so forth. Different aggregation mechanisms require different resources and will therefore influence the routing strategy such as adapting the routing to the application or vice versa. Decision must be made as to whether query processing is to be performed at sensor nodes or only at designated, resource rich nodes, placing simple filters at peripheral sensing nodes. Typical aggregation operation include MAX, MIN, AVG, SUM and many other well-known database management techniques.

There are two types of addressing in sensor network: data centric and address centric. In a data centric paradigm, a query will be sent to specific region in the network, whereas in addressing centric, the query will be sent to an individual node. A user may want to query an individual node or a group of nodes for information collected in the region. Depending on the amount of data fusion performed, it may not be feasible to transmit a large amount of the data across the network. Instead, various local sink nodes will collect data from a given area and create summary messages. A query may be directed to the sink node nearest to the desired location.

Once a specific event is detected, it needs to be delivered to subscribers. Because of the overlap in the proximity ranges of sensors, the same phenomena might be recorded by multiple sensor nodes. Alternatively, systematic aggregation might lose all the data on the same phenomenon. End-to-end event transfer schemes that fit the characteristics of WSNs are needed, in the same way that delivery semantics of asynchronous communication, such as publish/subscribe, is needed for distributed systems. Because energy is limited by the capacity of battery, it is necessary to consider the extend to which the demands of applications can be met. Adaptive communication protocols, e.g., power aware protocols, adaptive middleware, are actively being researched.

1.3.4 Self Configuration, Real-Time, and Reliability

Given the large number of nodes and their scattered placement in hostile locations, it is essential that the network be able to self-organize. Moreover, nodes may fail from limitation of energy, from physical destruction or any other means, and new nodes may need to join the network. The nodes can also coordinate to exploit the redundancy provided by high density as to extend the overall system lifetime. The large number of nodes deployed in the system will preclude manual configuration, and the environmental dynamics will preclude design-time pre-configuration. Nodes will have to self-configure to establish a topology that provides communication under stringent energy constraints. The network must be able to continuously and periodically reconfigure itself so that it can continue to function properly and serve its purpose. A high degree of connectivity is therefore essential.

The reliability $R_k(t)$ or fault tolerance of a sensor node is modeled in [11] using the Poisson distribution to capture the probability of not having a failure within the time interval (0; t):

$$R_k(t) = \exp(-\lambda_k t)$$

where λ_k and t are the failure rate of sensor node k and the time period, respectively. The fault tolerance level depends on the application level requirement of the WSN.

Applications like object tracking may need to correlate events from different source nodes in real-time. Real-time support (e.g. a physical event must be reported within a certain period of time) may be critical in WSNs. This aspect affects time synchronization, scheduling, and power saving algorithms, which may be affected by the network topology and the communication mechanism.

1.4 Organization of Report

In this thesis, we deal with data processing in sensor network primarily. In particular, data dissemination and aggregation problems are addressed and novel yet energy efficient solutions are sought. In addition, sensor data calibration problem is also considered.

Chapter 2 gives an overview of all the background concepts involved in this thesis. The fundamentals of gossip algorithms are introduced and elaborated in detail. After which, the network and timing models used throughout the thesis is presented, and finally a brief discussion on network coding is given.

Chapter 3 and 4 presents our solutions for two well-known problems in data-centric wireless sensor networks: information spreading and data aggregation. These include the problem abstractions, the detailed mathematical analysis and proof, as well as simulation results and thorough discussions.

Chapter 5 addresses a slightly different problem: sensor data calibration. The formal definition of measurement error and various types of calibration techniques are introduced. Moreover, the rationale and steps of applying loopy belief propagation are discussed. Lastly, the performance on a real sensor network is given.

Chapter 6 gives a short conclusion and recommendations for future work.

Chapter 2

Preliminaries and System Models

In this chapter, we will introduce the basic gossip algorithm first, followed by the network model, time model used throughout the report. Finally, a brief explanation on network coding is given.

2.1 Basics of Gossip Algorithm

The recent advances in peer-to-peer, mobile ad-hoc and wireless sensor networks have triggered the design of robust, simple, scalable and energy efficient information exchange algorithms. Gossip based algorithms for information dissemination have recently received significant attention for sensor and ad-hoc network applications due to their simplicity and robustness. Consider a network of n nodes, in which each node u_i has a piece of information m_i, and the objective is to disseminate all information among all nodes in the network quickly and cheaply. The use of gossip algorithm to solve this kind of problem was first proposed by Demers *et al* [12]. They considered a database which is replicated at many sites, and the objective is to maintain mutual consistency among the sites in the face of updates.

In particular, they decomposed the update procedure into two steps. At first, every site chooses another site at random and the two sites exchange with each other their complete database contents. After this, once a site receives new updates, it becomes a hot rumor and periodically updates other sites randomly. Since this paper, gossip algorithm has became an interesting topic for many researchers. Kempe *et al* [20] summarized two fundamental design aspects for any gossip-based protocols. The first is, in each round, every node selects its communication partner, either in a deterministic or random fashion. The other important issue is to decide the content of the message to send to the communication partner.

In each round, the communication graph is generated in the following manner. Each node $u \in \{u_1, \ldots, u_n\}$ determines a communication partner v uniformly and randomly

from the same set of sensor nodes. There are two models of message transmission once the partner is determined. In **PULL** model, a message is transmitted from a called node to the caller node, i.e., from node v to u. While for the **PUSH** model, the message is transmitted from the caller node to the called node. At a first glance, the two models seem to be the same. However, in **PULL** model, each node will receive one message from other nodes and not every node is necessary to transmit. In contrast, for **PUSH** model, each node will transmit one message to other nodes and not every node is necessary to receive. For both models, only one message can be transmitted between the selected communication pair. Figure 2.1 shows the communication partners selected for a 16 nodes network.

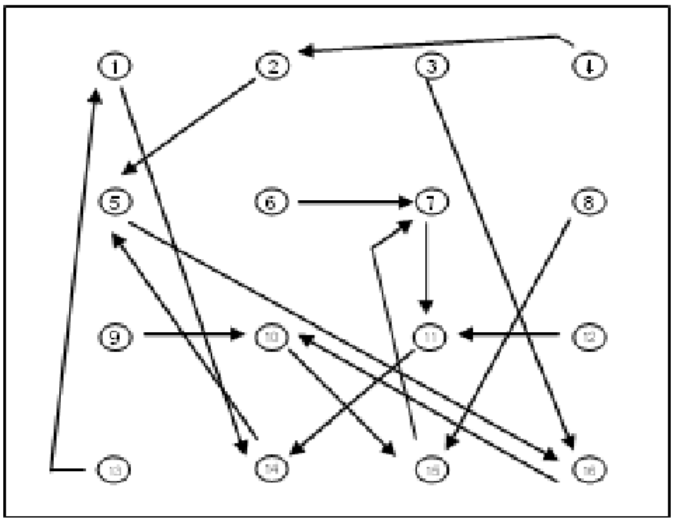

Figure 2.1: Communication partner selection.

The basic gossip algorithm employs a random message selection strategy to determine the message to be transmitted. This is a simple scheme, where the transmitting node simply looks at the message it has received so far and picks up any of the message uniformly and send to the receiving node. For instance, under **PUSH** model, M_u is the set of messages currently at node u, then u randomly selects a message e from M_u and transmits to v, where

$$P_r(e = m_i) = \frac{I(m_i \in M_u)}{|M_u|} \tag{2.1}$$

The $I(m_i \in M_u)$ is a function which counts the number of m_i in M_u and $|M_u|$ is the size of the message set.

2.2 System Models

The following network and time models are widely used in sensor networks and provide good abstraction for high-level protocols/algorithms development.

2.2.1 Network Model

To model a wireless ad-hoc or sensor network, a connected graph $G = (V, E)$ is considered, where the vertex set V corresponds to n nodes and E is the set of connectivity edges. The graph is assumed to operate under the random geometric graph (RGG). This model was first introduced by Gupta and Kumar [33] and is a popular model for sensor networks. A d-dimensional RGG is a graph where each of the n vertices is assigned random coordinates in the box $[0, 1]^d$, and only points "close" to each other are connected by an edge. The degree distribution of a RGG with average connectivity α can be modeled by a binomial function. RGGs are sometimes referred as spatial graphs [16, 17]. Figure 2.2 illustrates a RGG in two dimensions (2D). There are n sensor nodes randomly and uniformly placed in a 2-dimensional field. Two nodes are connected to each other if they are within transmission radius r, i.e., there is a bidirectional edge between them.

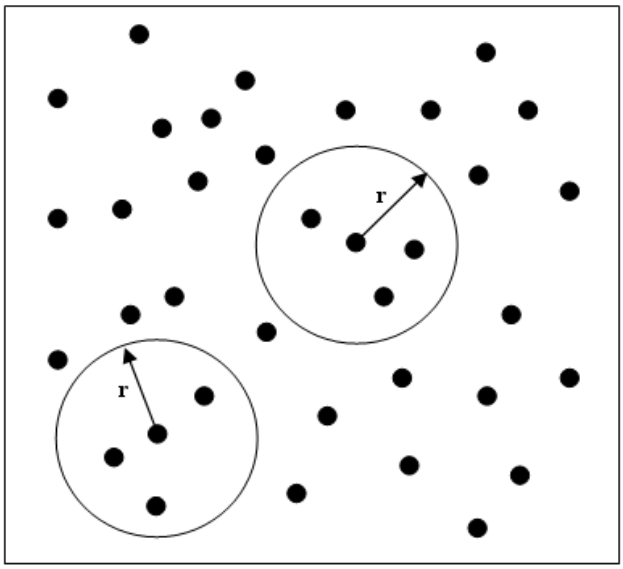

Figure 2.2: Network topology model.

The volume of a d-dimensional (hyper) sphere with radius r is:

$$V_{sphere} = \frac{\pi^{d/2} r^d}{\Gamma(\frac{d+2}{2})} \tag{2.2}$$

where $\Gamma(x)$ is the gamma function. This volume is needed in order to find the edges in RGGs. To visualize a RGG in general, one can think of box filled with small spheres with radius r and volume V given by the above equation, where the points are connected by an edge only if the distance between their center is $< 2r$, i.e., if the spheres overlap. Since the total volume of our box is 1, the probability that any two arbitrarily chosen vertices are connected is equal to the volume of a sphere with radius $R = 2r$. In continuum percolation theory this volume is denoted as the excluded volume V_{ex}, where $V_{ex} = 2^d V$ in a RGG. The excluded volume is the basic quantity of interest because it directly related to the connectivity.

2.2.2 Time Model

There are two models widely used for gossip algorithm, namely asynchronous time model and synchronous time model. The qualitative and quantitative conclusions are unaffected by the type of model [18]. The choice of model is entirely based on convenience.

- **Asynchronous time model**: Each node has a clock which ticks at rate 1 Poisson process [19]. Thus, the inter-tick times at each node are rate 1 exponentials, independent across nodes and over time. Equivalently, this coresponds to a single clock ticking according to a rate n Poisson process at time Z_k, $k \geq 1$, where $\{Z_{k+1} - Z_k\}$ are IID exponentials of rate n. Let $I_k \in \{1, \ldots, n\}$ denote the node whose clock ticked at time Z_k. Clearly, the I_k are IID variables distributed uniformly over $\{1, \ldots, n\}$. We discretize time according to clock ticks since these are the only times at which the value at each node changes. Therefore, the interval $[Z_k, Z_{k+1}]$ denotes the k^{th} time-slot and, on average, there are n clock ticks per unit of absolute time.

- **Synchronous time model**: In the synchronous time model, time is assumed to be slotted commonly across nodes. In each time slot, each node contacts one of its neighbors independently and (not necessarily uniformly) randomly. Note that in this model all nodes communicate simultaneously, in contrast to the asynchronous time model where only one node communicates at a given time. On the other hand, in both models each node contacts only one other node at a time.

2.3 Network Coding

Communication networks today share the same fundamental principles of operation. Whether it is packet over the Internet, or signals in a phone network, information is transported in the same way as cars share a highway or fluids share pipes. That is, independent data streams may share network resource, but the information itself is separate.

Routing, data storage, error control, and generally all network functions are based on this assumption.

Network coding [13] is a recent field in information theory that breaks the traditional routing concept. Instead of simply forwarding data, nodes may recombine several input packets into one or several output packets. A simple example in a wireless context is a three node topology, as shown in Figure 2.3. BS1 and BS2 are located at a distance twice the wireless transmission range. Installed at the middle is a relay transceiver RL. Under this setting, each physical-layer transmission consumes a unit amount of energy. It is easy to see that the minimum amount of energy required to exchange b_1 and b_2 between BS1 and BS2 is four (transmissions) using the conventional routing approach. One such solution is: BS1 transmits b_1 to RL, RL forwards b_1 to BS2. BS2 transmits b_2 to RL, RL forwards b_2 to BS1. With network coding on this example, only three transmissions are necessary. First, BS1 transmits b_1 to RL, and BS2 transmits b_2 to RL. Next, RL broadcasts the XOR result of b_1 and b_2, $b_1 \oplus b_2$, to BS1 and BS2 using only one transmission. BS1 and BS2 can recover both b_1 and b_2 by each solving a simple linear system of equations. Network coding allows for a much larger degree of flexibility in the way packets can be transmitted in the intermediate nodes. In addition to the throughput benefits evidenced in this example, network coding is also very well suited for environments where only partial or uncertain information is available at decision making. Similar to erasure coding, successful reception of information does not depend on receiving specific packet content but rather on receiving a sufficient number of independent packets. The topic of network coding is interdisciplinary in nature, involving graph theory, algorithms, information theory, coding theory, optimization theory, etc. The network coding page [14] provides a bibliography of papers related to network coding. Therefore, in this thesis we will only explain what is linear network coding and how it can be implemented.

2.3.1 Linear Network Coding

Consider a system that acts as information relay, such as a router, a node in an ad-hoc or peer to peer network. Traditionally, when forwarding an incoming packet destined to some other nodes, it simple repeats it on one or more of its outgoing links. With network coding, we allow the node to combine a number of packets it has received or created into one or several output packets. Let us suppose each packet is of length l bits, when the packets to be combined do not have the same size, the shorter ones are padded with trailing 0s. We can interpret q consecutive bits of a packet as a symbol over the field F_2^q, with each packet consisting of a vector of l/s symbols. With linear network coding, outgoing packets are linear combinations of the original packets, where addition and multiplication are performed over field F_2^q. The reason for choosing a linear

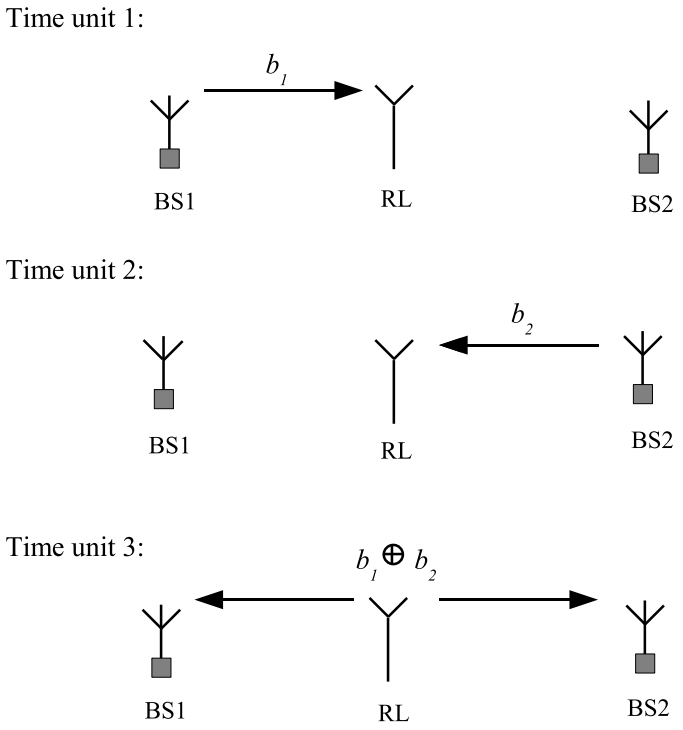

Figure 2.3: Information exchange between two wireless base stations (BS1 and BS2) through relay transceiver (RL).

framework is that the algorithm for coding and decoding are well understood and easily implementable.

Linear combination is not concatenation: if we linearly combine packets of length l, the resulting encoded packet also has size l. In contrast to concatenation, each encoded packet contains only a fraction of the information contained in the original packets. One can think of linear network coding as a form of information spreading.

2.3.2 Encoding and Decoding

Assume that a number of original packets $M^1, \ldots, M^n$ are generated by one or several sources. In linear network coding, each packet through the network is associated with a sequence of coefficients $g_1, \ldots, g_n$ in F_2^q and is equal to $E = \sum_{i=1}^{n} g_i M^i$. The summation has to occur for every symbol position, i.e., $E_j = \sum i = 1^n g_i M_j^i$, where M_j^i and E_j is the j^{th} symbol of M^i and E respectively. For a three symbol example, M^1, M^2, M^3, the coded packet E can be $4M^1 + 5M^2 + 7M^3$. Please notice that all operations are performed over the finite field F_2^q.

Based on whether the coded packet contains the coefficient vector, linear network coding scheme can be broadly classified into fixed network coding and random network

coding [15]. Under random network coding, both the coefficient vector $g = (g_1, \ldots, g_n)$ (encoding vector) and the encoded data $E\sum_{i=1}^{n} g_i M^i$ are contained in the same packet. For example, the encoding vector $g = (1, 0, \ldots, 0)$, where only the first coefficient is 1, means the information is simply the first packet M^1. On the other hand, for fixed network coding, the encoding vectors have been pre-allocated and are entirely dependent on the network topology. Once topology changes, certain encoding vectors may no longer be valid, and refreshment of the vectors are required. Since sensor network is expected to have frequent topology updates, we shall discuss random network coding in detail in the next paragraph.

We can represent the network topology as a directed graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. An edge is denoted as a tuple $e = (u, v) \in E$, where $u = tail(e)$ and $v = head(e)$. The set of edges terminating at a particular node $u \in V$ is represented as $\Gamma_I(u) = \{e \in E : head(e) = u\}$. And similarly $\Gamma_O(u) = \{e \in E : tail(e) = u\}$ denotes the set of edges originating from node $u \in V$. Without loss of generality we assume that each edge $e \in E$ has unit capacity, i.e., node u can send information through edge $e = (u, v)$ at 1 symbol per time unit. We model links with larger capacities as multiple edges connecting two nodes. The capacity $C_G(s, t)$ of network G is the maximal possible information rate from node s to t.

Discrete random processes $X_1, \ldots, X_k$ are observable at the source node, and each of them can be represented by one symbol. To simplify the analysis, the communication links are assumed to be free of delays and losses. Under random linear network coding, edge $e = (u, v)$ carries the process $Y(e)$, which is defined as:

$$Y(e) = \sum_{i:X_i \text{ at } u} \partial_{i,e} X(u, i) + \sum_{e':head(e')=u} \beta_{e',e} Y(e') \tag{2.3}$$

where ∂ and β are random combination coefficients in network coding, and $X(u, i)$ is the i^{th} random process at node u. Since there is only one source node in the network, $X(u, i) = 0$ for all $u \neq s$. By solving (2.3), the random processes at edge set E can be represented as:

$$Y(E) = XA(I - F)^{-1} \tag{2.4}$$

where X is the input vector at source node and is of size $1 \times k$, I is the identity matrix, A and F represent the linear mixing of the input vector and the adjacency matrix, and have size $k \times |E|$ and $|E| \times |E|$, respectively.

Assume a node has received the set $(g^1, E^1), \ldots, (g^n, E^n)$. In order the retrieve the original packets, it needs to solve the n linearly independent equations where the unknowns are the M^is. This is a linear system with exact n equations and n variables. In reality, there could be more equations, i.e., $\geq n$ equations, to solve the n unknowns.

However, this set of equations must satisfy the following condition: there must be n linear independent equations. This is a major advantage of network coding, the recipient node only needs to collect that much number of packets regardless of the order and packet sequence number in that matter.

Chapter 3

NBGossip: Neighborhood Gossip with Message Aggregation by Network Coding

Gossip based algorithms for information dissemination have recently received significant attention for sensor and ad-hoc network applications due to their simplicity and robustness. However, a common drawback of many gossip based protocols is the waste of energy in passing redundant information over the network. Thus gossip algorithms need to be re-engineered in order to be applicable for energy constrained networks. In this Chapter, we consider a scenario where each node in the network holds a piece of information (message) at the beginning, and the objective is to simultaneously disseminate all information (messages) among all nodes quickly and cheaply. To provide a practical solution to this problem for ad-hoc and sensor networks, *NBgossip* algorithm is proposed, which is based on network coding and neighborhood gossip. In NBgossip, nodes do not simply forward messages they received, instead, the linear combinations of the messages are sent out. In addition, every node exchanges messages with its neighboring nodes only. Mathematical proof and simulation studies show that the proposed NBgossip terminates in the optimal $O(n)$-order rounds and outperforms the existing gossip based approaches in terms of energy incurred to spread all information.

3.1 Related Work

During the past two decades, gossip-based algorithms have been studied in great details. Karp *et al* [21] have shown in their work that gossip algorithms can not perform better than a sequential approach of spreading the messages one after the other in terms of propagation rounds. This yields a $\Theta(n \ln n)$ performance bound for gossip algorithms. [22] proposed *spatial gossip* algorithm and proved that new information can be spread

to nodes at distance d with a high probability in $O(\log^{1+\epsilon} d)$ rounds. In spatial gossip, instead of uniformly selecting the communication partners in each round, the nearby nodes are selected with a higher probability compared to further located nodes. Even though the algorithm was originally devised to disseminate new information with a delay growing slowly with d and independent with network size n, it is observed that it can also save the energy incurred during information spreading. However, a common drawback of gossip algorithms is the waste of energy in passing redundant information around in the network.

To mitigate the drawback mentioned above, while using gossip algorithms to solve the averaging problem, Boyd *et al* [18] found that the number of rounds or averaging time is closely related to the mixing time of the Markov chain defined by a weighted random walk on the graph. Furthermore, they proposed a subgradient method to optimize the neighbor selection probabilities for each node in order to find the fastest-mixing Markov chain on the graph. This helps to reduce the total number of rounds to compute the average. Later on, [23] further reduced the number of message transmissions in wireless sensor networks by exploiting geographical information in sensor network. In spite of the progress in using gossip algorithms to solve *Nodes Aggregation* [24] problems, the results are not directly applicable to information spreading problem. Until very recently, [25] introduced random network coding into gossip based algorithms and proposed algebraic gossip (*alge-gossip*). In each round of alge-gossip, rather than randomly choosing one piece of data it received so far, each node linearly combines all the received data and sends to its communication partner. By adopting this coding approach, the information can be exchanged in $O(n)$ rounds with a large probability. Unfortunately, alge-gossip is not energy efficient as explained later in Section 3.5. In addition, [26] employed a multi-message store-and-forward approach and demonstrated that their protocol can spread k initial messages among n participants in $O(k + \ln n)$ rounds with high probability. However, the underlying assumption of their approach is the presence of a complete communication graph, which is extremely hard to maintain in wireless ad-hoc and sensor networks.

Network coding [27] [28] is a recent field in information theory that breaks the traditional routing concept. Instead of simply forwarding data, nodes may recombine several input packets into one or several output packets. There are two main benefits of this approach: potential throughput improvements and a high degree of robustness [13]. Li *et al* [29] pointed out that linear network coding can achieve the min-cut boundary in networks with multicast connections. [15] and [30] presented the idea of random network coding, which ensures that the linear coding operation at each node is fully distributed. By introducing network coding into gossip based protocols, we can avoid the situation that a node may receive a message it already owns, and this helps to reduce the number of rounds significantly to disseminate all information.

3.2 Uniform Gossip, Spatial Gossip, and Alge-gossip

As suggested in [20], any gossip protocols can be decomposed into two steps. The first step is communication partner selection. In every round, each node selects its communication partner, either in a deterministic or random manner. The second step is message transmission. After the communication partner is fixed for a node, it decides the content of the message to send out. In this part, we shall describe uniform gossip [12], spatial gossip [22] and alge-gossip [25] based on the two steps. The notations used throughout this Chapter are summarized in the following table:

Table 3.1: Summary of Notations

Notation	Definition
N	the set of sensor nodes in the network
n	number of sensor nodes
u	a certain node in the network
m	a message in general
u_i	the i^{th} sensor node
m_i	the message with sensor node u_i in the beginning
M_u	the set of messages received so far at node u
M_{u_i}	the set of messages received so far at node u_i
M	message set $\{m_1, m_2, \cdots, m_n\}$
e	message sent out from a sensor node
F_q	finite field of size q, $q \gg n$
F_q^h	h-dimensional finite space

3.2.1 Uniform Gossip

- Communication Partner Selection: Each node $u \in N = \{u_1, u_2, \ldots, u_n\}$ determines a partner v uniformly and randomly from the same set N.

- Message Transmission: Node u looks at the messages it has received so far and picks up one of them randomly to send to node v. The probability for a message m_x to be sent out is:

$$Pr(e = m_x) = \frac{I(m_x \in M_u)}{|M_u|}, 1 \leq x \leq n \tag{3.1}$$

  The $I(m_x \in M_u)$ is a function which counts the number of m_x in M_u, and $|M_u|$ is the size of the set.

3.2.2 Spatial Gossip

- Communication Partner Selection: Instead of uniformly selecting a partner from the set N, each node chooses nearby nodes with a higher probability than further

located nodes. The selection is based on an inverse-polynomial probability distribution. Let us denote the distance between node u and v as d, then the probability for u to pick up v is:

$$p_{u,v} = c_x(d+1)^{-2\rho} \tag{3.2}$$

where c_x is the normalization constant and ρ is the exponent coefficient with $1 < \rho < 2$.

- Message Transmission: It follows the same procedure as in uniform gossip.

3.2.3 Alge-gossip

Recognizing the shorting coming of traditional gossip, Alge-gossip combines network coding with message transmission. Instead of randomly selecting a message from its message pool, network node linearly combines all messages available and transmits the encoded message to its communication partner. Therefore, when a node receives an encoded message, it is unlikely that the new message is duplicate. By employing this strategy, Alge-gossip avoids transmitting redundant information across the network.

- Communication Partner Selection: The partners are selected uniformly and randomly as in uniform gossip.

- Message Transmission: This part differs significantly from the above two gossip algorithms. With network coding, the messages are treated as algebraic entities on which arithmetic operations can be performed. Let us assume that a message m is of size l bits, and m can be viewed as a $h = \lceil l/\log_2(q) \rceil$-dimensional vector over F_q. All arithmetic operations are performed over the finite field, as a result, it guarantees that the coded messages are recoverable. At the start of a particular round, a message at node u, $f_j \in M_u$, $j = 1, 2, \cdots, |M_u|$, can be written as:

$$f_j = \sum_{x=1}^{n} \partial_{jx} \cdot m_x \tag{3.3}$$

where $\partial_{jx} \in F_q$. Now, suppose node u starts to transmit a message e to v, where e is expressed as:

$$e = \sum_{f_j \in M_u} \beta_j \cdot f_j \tag{3.4}$$

The coefficients β_js are randomly selected from F_q. By substituting (3.3) into (3.4), the coded message e is as follows:

$$e = \sum_{x=1}^{n} \gamma_x \cdot m_x \tag{3.5}$$

where

$$\gamma_x = \sum_{j=1}^{|M_u|} \beta_j \cdot \partial_{jx} \tag{3.6}$$

and $\gamma_i s$ are transmitted together with e as a small overhead.

3.3 NBgossip

In this part, we present the main idea of NBgossip, which combines neighborhood gossip and network coding. Based on the network model introduced in Chapter 2, the nodes are distributed in the operation area with uniform density. Which is equivalent to say that within communication radius r, each node is assumed connection to exactly k neighboring nodes. In order to have good connectivity and to minimize interference, r has to be in the scale of $\Theta(\sqrt{\frac{\log n}{n}})$. Consequently, k would be in the order of $\Theta(\log n)$. It is assumed that the total number of nodes in then network is known in a prior.

3.3.1 NBgossip Algorithm

Here we introduce a new term called *super round*, which is denoted as S. Each super round consists of k rounds, i.e. system advances from super round S to $S+1$ after k rounds have passed. Let e_{ji}^S denote the coded message received by node u_i from its j^{th} neighbor in super round S, with $1 \leq j \leq k$. In addition, we use $E_i^S = (e_{1i}^S, e_{2i}^S, \cdots, e_{ki}^S)^T$ and $E_i = (E_i^1, E_i^2, \cdots, E_i^S)^T$ to represent the coded messages received by u_i at super round S and all the coded messages received so far by u_i, respectively.

Since only linear operations are performed over the message set M, we can write e_{ji}^S in the form of $V_{ji}^S \cdot M^T$, where V_{ji}^S is a $1 \times n$ vector with each element from F_q. For example, if $n = 3$ and the message $e = m_1 + 2 \cdot m_2 + 4 \cdot m_3$, e can be written in the form of $(1, 2, 4) \cdot (m_1, m_2, m_3)^T$. Finally, $C_i^S = (V_{1i}^S, V_{2i}^S, \cdots, V_{ki}^S)^T$ and $C_i = (C_i^1, C_i^2, \cdots, C_i^S)^T$ are used to denote the current coefficient matrix received by u_i in super round S and the coefficient matrix received so far by u_i, respectively. It should be noted that C_i is of size $kS \times n$. With these notations introduced, the pseudocode for NBgossip is presented in Algorithm 1.

The e_{ji}^0 is initialized to m_i and the V_{ji}^0 is initialized to be the i^{th} base vector of n-dimensional finite space, i.e., the vector with 1 in the i^{th} position and 0 in all others. Please note that all arithmetic operations are defined over F_q. In each super round S, node u randomly combines all the coded messages received in the previous super round except the one originated from the j^{th} neighbor. Afterwards, u sends the resulted message to the very j^{th} neighbor. All together k such messages are sent out from u to its k neighbors. Meanwhile, u would have received k messages from these neighbors in

Protocol 1 Neighborhood Gossip in Wireless ad-hoc and Sensor Networks

1: **while** for all nodes u_i, Rank$(C_i) \neq n$ **do**
2: System advances from super round S to $S+1$
3: **for** round $j = 1 : k$ within super round S **do**
4: Each node u_i composes the coded message e for its j^{th} neighbor, where $e = \sum_{x=1,x\neq j}^{k} \beta_x \cdot e_{xi}^S$ and β_x is randomly drawn from F_q.
5: The coefficient vector $V = \sum_{x=1,x\neq j}^{k} \beta_x \cdot V_{xi}^S$ is appended to e and transmitted together to the j^{th} neighbor of node u_i.
6: **end for**
7: Meanwhile, node u_i receives k coded messages from its neighbors. Let us assume that coded message $\overline{e_l}$ and coefficient vector $\overline{V_l}$ are from one of the k neighbors, say l^{th} neighbor.
8: Update $e_{li}^{S+1} = \overline{e_l}$, $V_{li}^{S+1} = \overline{V_l}$, $l = 1, \cdots, k$.
9: **end while**
10: Solve the n variable equation $C_i \cdot M^T = E_i$ and obtain the message set M.

exactly the same super round. This process repeats until all nodes have obtained the message set M.

Based on the protocol description, all nodes will obtain their full rank coefficient matrix simultaneously. However, in reality, some node u might attain full rank first. It will exit from the message exchanging cycles immediately, i.e., no more coded message and coefficient vector will be pushed into E and C. Nevertheless, the node would still be assisting neighboring nodes to gather all the information. To do so, it will first advertise its own status of completion to its one hop neighbors. Once the completion status message is heard, neighboring nodes would stop sending any message to u. Meanwhile, u continues to transmit messages to its neighbors as described in the protocol. Instead of combining messages received from neighbors in the previous super round, u simply combines all the coded messages received so far in a random fashion. Since u is of full rank already, the coded coefficient vectors generated by u can be considered as independently and randomly drawn from the $n-$dimensional finite space, which ensures the correctness of the protocol.

3.4 Mathematical Analysis of NBgossip

When a node attains its full rank coefficient matrix, it is able to compute the message set M. Thus, to assess the number of rounds required, we only need to study the increment of the rank of the coefficient matrix. If it can be proved that each newly

received message increases the rank of coefficient matrix with high probability (the claim holds with probability at least $1 - O(\frac{1}{n})$), the protocol would terminates in $O(n)$ rounds. We define R_i to be the minimum number of steps needed for node u_i to visit any other nodes based on neighborhood connections, and R to be the maximal of R_i, where $1 \leq i \leq n$. First, we shall prove that the coefficient matrix at node u_i has some all-zero columns before R_i super rounds are passed, and each newly received coefficient vector would have at least one non-zero value at the positions of the all-zero columns with high probability. Which means, the new coefficient vector will increase the matrix rank by at least one. Once R_i super rounds have passed, by establishing the fact that any coefficient vectors received by u_i in super round S can be written as a linear combination of all coefficient vectors generated in super round S' where $S - S' \geq R_i$, we show that the coefficient vector received by u_i will increase the rank of C_i with high probability. It is followed by proving that R is in the order of $\Theta(\sqrt{\frac{n}{\log n}})$. Finally, we evaluate the lower and upper bounds on the number of rounds needed to spread all the information.

Theorem 3.1 *The probability that the coefficient vector from node u_i to its j^{th} neighbor in super round $S+1$ attains the maximal possible non-zero columns from linear combination of V_{xi}^S is at least $(1-1/q)^n$, where $x \neq j$ and $1 \leq x \leq k$.*

Proof: If the y^{th} columns of all V_{xi}^S are zero, where $x \neq j$, then the y^{th} column of V_{ij}^{S+1} must be zero. Otherwise, if any of them are non-zero, then the y^{th} column of the randomly combined coefficient vector is not zero with probability $(1 - 1/q)$. Let us represent the value at the y^{th} column of V_{xi}^S with α_x, and the random coefficient chosen for V_{xi}^S using β_x. The following equation must be satisfied in order for the y^{th} column to be zero:

$$\sum_{x=1, x \neq j}^{k} \alpha_x \cdot \beta_x = 0 \tag{3.7}$$

Since (3.7) represents a hyperplane within h (number of non-zero α_x) dimensional space, and there will be q^{h-1} vectors contained in this hyperplane. Compared to the total number of q^h vectors in the h-dimensional space, the possibility for (3.7) to hold is $1/q$. All together there are at most n columns that are not all-zero, so the probability for V_{ij}^{S+1} to achieve the maximal non-zero columns is at least $(1 - 1/q)^n$.

Before R_i super rounds are reached, there exists a certain node u_w such that $dist(u_i, u_w) = dist(u_j, u_w) + 1$, where $dist(u, v)$ is the least number of steps from node u to v based on neighborhood connections, and u_j is one of the neighbors of u_i. It is not hard to see that the w^{th} column of C_i at node u_i will be all-zero before $dist(u_i, u_w)$ super rounds. In super round $dis(tu_j, u_w)$, some coefficient vectors received by u_j would have a non-zero value at the w^{th} position with high probability. In super round $dist(u_i, u_w)$, u_j randomly

combines all its coefficient vectors and sends the result vector to u_i. Based on Theorem 3.1, this random combination maintains the non-zero value at the w^{th} column of this result vector with a probability of at least $(1-1/q)^n$. Hence on receiving this vector, the probability that the coefficient matrix C_i at u_i would increase its rank is greater than $(1-1/q)^n$.

Theorem 3.2 *Once R_i super rounds have passed, the coefficient vector V_{ji}^S at super round S can be written as a linear combination of all the coefficient vectors generated in super round S', i.e., $\sum_{p=1}^{p=n}\sum_{x=1}^{x=k}\beta_{xp}\cdot V_{xp}^{S'}$ provided that $S-S'\geq R_i$, where β_{xp} is a random number.*

Proof: Since the messages are exchanged recursively, the communication graph can be modeled by introducing another dimension - super round (time). To perform the mapping, the vertices in each layer correspond to the set of coefficient vectors at each super round. For example, V_{ji}^S represents the coded vector received by node u_i from its j^{th} neighbor in super round S. Based on the protocol description, each coefficient vector V_{ji} in super round S is a linear combination of some coefficient vectors in super round $S-1$. If V_{xy}^{S-1} happens to be such a vector, there would be an edge connecting from V_{ji}^S to V_{xy}^{S-1}. In addition, the weight of the edge corresponds to the random combination coefficient β for V_{xy}^{S-1}. By repeating this process, the communication graph shown in Figure 3.1 can be obtained.

Since R_i is the least number of steps required for node u_i to visit any other nodes based on neighborhood connections, V_{ji}^S could reach any coefficient vector in layer S' by walking along the arcs in the communication graph provided that $S-S'\geq R_i$. By reversing the arcs and multiplying the weights along the edges from $V_{xp}^{S'}$ to V_{ji}^S, it is not hard to see that V_{ji}^S can actually be written as: $\sum_{p=1}^{p=n}\sum_{x=1}^{x=k}\beta_{xp}\cdot V_{xp}^{S'}$.

Before we move on to assess whether V_{ji}^S would increase the rank of C_i for $S\geq R_i$, the following two theorems are introduced.

Theorem 3.3 *Let us denote the global coefficient matrix for super round S as $G^S=(C_1^S,C_2^S,\cdots,C_n^S)^T$, then G^S preserves its rank n in each and every super round S with a probability greater than or equal to $(1-1/q)^{(k-2)}$.*

Proof: In each super round, the messages exchanged between neighbors are defined as a linear combination of the messages received in the previous super round. Since only linear operations are involved, we can define a global transformation matrix T_r such that $T_r\cdot G^S=G^{S+1}$. Based on the understanding of Algorithm 1, we can use (3.8) to express the transformation equation.

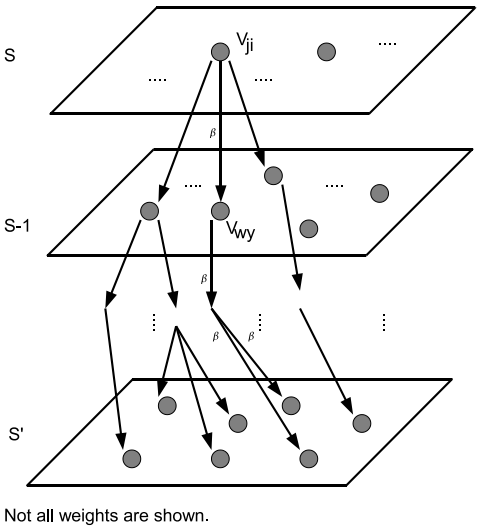

Figure 3.1: Time sliced communication graph.

$$\begin{pmatrix} \mathbf{0} & \dots & 0\ \beta_{l12}\dots\beta_{l1k} & \dots & \mathbf{0} \\ \mathbf{0} & \beta_{p11}\ 0\dots\beta_{p1(k-1)}\ \beta_{p1k} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \beta_{121}\ 0\ \beta_{123}\dots\beta_{12k} & & \mathbf{0} & \dots & \mathbf{0} \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix} \cdot \begin{pmatrix} V_{11}^S \\ V_{21}^S \\ \dots \\ \dots \\ V_{kn}^S \end{pmatrix} = \begin{pmatrix} V_{11}^{S+1} \\ V_{21}^{S+1} \\ \dots \\ \dots \\ V_{kn}^{S+1} \end{pmatrix} \tag{3.8}$$

$$\begin{pmatrix} A_1^k & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & A_2^k & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & A_n^k \end{pmatrix} \cdot \begin{pmatrix} V_{11}^S \\ V_{21}^S \\ \vdots \\ V_{kn}^S \end{pmatrix} = \begin{pmatrix} V_{u_1 \to 1^{st} neighbor} \\ V_{u_1 \to 2^{nd} neighbor} \\ \vdots \\ V_{u_n \to k^{th} neighbor} \end{pmatrix} \tag{3.9}$$

In T_r, the $\mathbf{0}$ is of size $1 \times k$ and equals to $(0, 0, \dots, 0)$. In order to understand how T_r is obtained, let us take the first row as an example. By following the matrix multiplication rule, we have $V_{11}^{S+1} = \sum_{x=1}^{k} \beta_{l1x} \cdot V_{xl}$, which is the coefficient vector received by node u_1 from its first neighbor u_l. Furthermore, node u_1 happens to be the first neighbor of node u_l, from the fact that $\beta_{l11} = 0$. It is obvious that there will be $(l-1)$ $\mathbf{0}$s on the left of $(0, \beta_{l12}, \dots, \beta_{l1k})$ and $(n-l)$ $\mathbf{0}$s on the right. Please note that C_i^S, G^S, and T_r are of size $k \times n$, $kn \times n$, and $kn \times kn$, respectively. Once we understand how T_r is obtained, it can be shown that T_r is of rank kn with a probability greater than $(1-1/q)^{(k-2)}$ (see Theorem 3.4). Since $q \gg n$ and k is in the order of $\Theta(\log n)$, we have proved that G^S preserves its rank n with a high probability, which is greater than $(1-1/q)^{(k-2)}$. $(Rank(G^0) = n)$.

Theorem 3.4 *The probability that T_r is of full rank kn is at least $(1-1/q)^{(k-2)}$.*

Proof: Based on the understanding of transformation matrix T_r, each row of T_r corresponds to the linear combination coefficients of C_i^S at some node u_i. All together there will be k rows in T_r corresponding to the k neighbors of u_i. By putting these k rows together we have the following submatrix T_{ri}:

$$\begin{pmatrix} & & \overbrace{\quad\quad\quad}^{(k\cdot(i-1)+1)^{th}\ldots(ki)^{th}} & & \\ \mathbf{0} & \ldots & (0\ \beta_{i12}\ \beta_{i13}\ldots\beta_{i1k}) & \ldots & \mathbf{0} \\ \mathbf{0} & \ldots & (\ \beta_{i21}\ 0\ \beta_{i23}\ldots\beta_{i2k}) & \ldots & \mathbf{0} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \mathbf{0} & \ldots & (\ \beta_{ik1}\ \beta_{ik2}\ldots\beta_{ik(k-1)}\ 0) & \ldots & \mathbf{0} \end{pmatrix}$$

where $\mathbf{0}$ is the all-zero vector of size $1 \times k$. By rearranging the transformation in this manner, (3.9) can be obtained. Please be aware that $\mathbf{0}$ is an all-zero matrix of size $k \times k$ in (3.9) and $V_{u_i \to j^{th} neighbor}$ represents the coefficient vector sent from u_i to its j^{th} neighbor during super round $S+1$. Without loss of generality, we take the first row of T_{ri} as an example. As shown in (3.9), if the first row is independent of the other $k-1$ rows in T_{ri}, it is definitely independent of all other rows in T_r. The probability for the first row to be independent of the other rows equals to the probability that the following randomly generated matrix A_i^k is of full rank k:

$$A_i^k = \begin{pmatrix} 0\ \beta_{i12}\ \beta_{i13}\ldots\beta_{i1k} \\ \beta_{i21}\ 0\ \beta_{i23}\ldots\beta_{i2k} \\ \ldots \\ \beta_{ik1}\ \beta_{ik2}\ldots\beta_{ik(k-1)}\ 0 \end{pmatrix}$$

where the βs are random numbers from F_q. We claim that the probability for $Rank(A_i^k) = k$ is at least $(1-1/q)^{k-2}$. Let P_k denote the probability that A_i^k is of full rank k. By exploiting the structure of A^k, it can be expressed as:

$$A_i^k = \begin{pmatrix} & & \beta_{i1k} \\ & & \beta_{i2k} \\ & A_i^{k-1} & \cdots \\ & & \beta_{i(k-1)k} \\ \beta_{ik1}\ \beta_{ik2}\ldots\beta_{ik(k-1)} & & 0 \end{pmatrix}$$

Suppose A_i^{k-1} is of full rank, then $(\beta_{ik1}, \beta_{ik2}, \cdots, \beta_{ik(k-1)})$ can be written as a linear combination of all the rows in A^{k-1}. We use V_1 to denote the first row of A^{k-1}, the following equation can be obtained:

$$(\beta_{ik1}, \beta_{ik2}, \cdots, \beta_{ik(k-1)}) = \sum_{x=1}^{k-1} \alpha_x \cdot V_x \tag{3.10}$$

If A_i^k does not attain its full rank, then the k^{th} row of A_i^k must be dependent on the previous $(k-1)$ rows of A_i^k. Hence the following condition must be satisfied:

$$\sum_{x=1}^{k-1} \alpha_x \cdot \beta_{ixk} = 0 \tag{3.11}$$

The probability for (3.10) to be true is $1/q$ as explained in Theorem 3.1. It is noted that even if A_i^{k-1} is not at its full rank, A_i^k could still attain its full rank. Hence we have $P_k > P_{k-1} \cdot (1 - 1/q)$. For $k = 2$, matrix A^2 is always of full rank. In conclusion, the probability for A^k to reach full rank is at least $(1 - 1/q)^{k-2}$. Thus T_{ri} is of rank k with a probability of at least $(1 - 1/q)^{k-2}$. Since each submatrix T_{ri} does not affect the rank of other submatrices, the probability that T_r is of rank kn is exactly the same as the probability that T_{ri} is of full rank.

As long as R_i super rounds have passed, any V_{ji}^S can be written as $\sum_{p=1}^{p=n} \sum_{x=1}^{x=k} \beta_{xp} \cdot V_{xp}^{S'}$ based on Theorem 3.2, where β_{xp} is a random number. As the rank of the global coefficient matrix $G^{S'}$ is equal to n with high probability (Theorem 3.3), V_{ji}^S can be considered as a random vector drawn from F_q^n. Now let us suppose that the current coefficient matrix rank at node u is p, which is less than or equal to n. In order to increase the rank of local coefficient matrix, the randomly drawn vector V_{ji}^S must not lie in the p-dimension subspace spanned by the coded vectors at node u. The total number of vectors in subspace p is q^p while the number of vectors in the full rank space is q^n. Thus each newly received coefficient vector V_{ji}^S will increase the rank at node u_i with probability $(1 - 1/q^{n-p})$. Thus in order for node u_i to obtain its full rank coefficient matrix, the number of rounds needed is $k \cdot R_i + (n - p)$ with high probability.

By applying geographic routing, at most $\Theta(\sqrt{\frac{n}{\log n}})$ steps are needed to travel from one node to any other node in the network [11]. Hence it can be concluded that R is also in the order of $\Theta(\sqrt{\frac{n}{\log n}})$. Conclusively, the NBgossip needs R super rounds and $(n - p)$ rounds, where $p = \min(Rank(C_i)), i = 1, \ldots, n$. Since R is in the order of $\Theta(\sqrt{\frac{n}{\log n}})$ and k in the order of $\Theta(\log n)$, the total number of rounds must be less than $k \cdot R + (n - p) = \sqrt{n \cdot \log n} + (n - p)$, which is less than $2n$. On the other hand, regardless of whether R super rounds have passed or not, every newly received coded message will increase $Rank(C_i)$ with high probability. Thus, the least number of rounds required to disseminate n messages will be exactly n. In either case, NBgossip can effectively disseminate all the information in $O(n)$ rounds. We can relax the uniform density condition by assuming a parameterized node density, with β_1 and β_2 as the lower and upper density bounds, respectively. It is proved that NBgossip still holds under parameterized density.

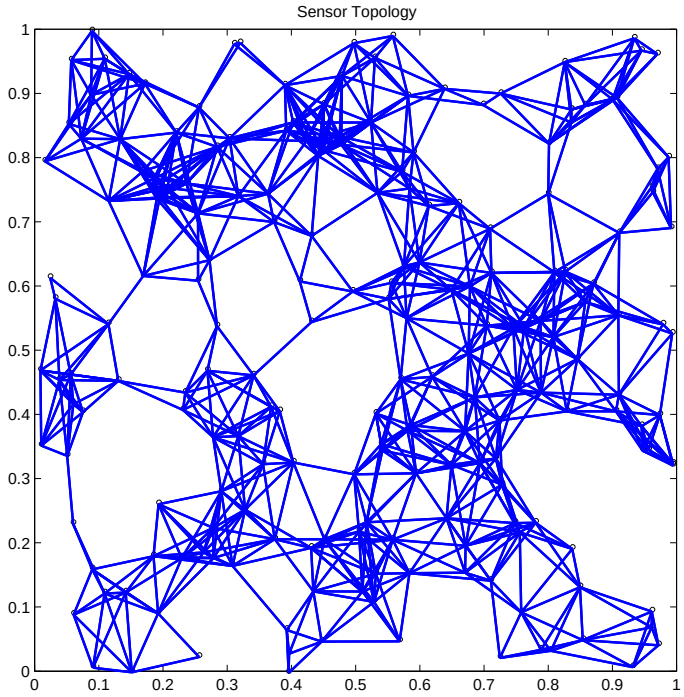

Figure 3.2: Sensor network topology.

3.5 Simulation Results and Discussion

In this section, the performance of NBgossip is compared with uniform gossip, spatial gossip, and alge-gossip in terms of number of rounds and energy consumed to disseminate all information. In addition, their respective computational complexity is evaluated.

3.5.1 Simulation Set Up

Based on previous works [16, 21], it is well known that in order to maintain good connectivity and minimize interference, the transmission radius r has to be scaled in the order of $\Theta(\sqrt{\frac{\log n}{n}})$. For the sake of simplicity, we assume that communication within transmission radius always succeeds. To start, a graph is generated randomly based on the geometric ensemble $G^2(n, r)$. By dividing the unit square into squares of length $(\sqrt{\frac{\log n}{2n}})$ and setting the transmission radius of each node r to be $(\sqrt{5\frac{\log n}{2n}})$, according to [11], following properties regarding the sensor nodes can be derived:

(i) *Each square contains at least one node.*

(ii) *Each node is connected to the nodes in the four adjacent squares.*

(iii) *Nodes in a same square are able to communicate with each other.*

Figure 3.2 shows the resulted geometric graph, with the solid lines representing graph connectivity. As long as the operation area is fixed, the number of neighboring nodes k would depend on the transmission radius r. It is known that r has to be in the order of $O(\sqrt{\frac{\log n}{n}})$, but the exact value of r to minimize the total energy consumption is beyond the scope of this thesis. In fact, r may not even necessarily be a fixed value. Moreover, this problem is more widely known as multi-source multi-casting network coding problem and is still an open research topic [10].

It is known that uniform gossip is not designed for energy constraint networks, but it is included here for completeness. In this network, each node has exactly one message and the objective is to disseminate all messages to all nodes. Since we are only concerned with the number of rounds and energy incurred for the four algorithms to terminate, only the coefficient vectors are transmitted across the network instead of the actual messages. It is clear that m_i can always be written in the form of the i^{th} base vector of n-dimensional finite space multiplies the message set M^T. Moreover, we define the cost in communication radius r to be unit energy consumption. For two nodes that are not within a single hop, the cost is defined to be the least number of hops to reach the other one. Finally, the cost between nodes are assumed to be symmetric.

3.5.2 Comparison of the Number of Rounds Needed

Figure 3.3 shows the mean completion time (in number of rounds) for the four gossip protocols. This mean is obtained by averaging the total number of rounds incurred over 20 executions. In addition, we conduct the simulation for network size from 9 nodes all the way to 100 nodes. Typically, WSN is scaled up to a few thousands. In previous section, we have proved the performance of NBgossip mathematically: NBgossip terminates in the optimal $O(n)$ rounds, where n is the number of sensor nodes. Here the simulation only serves to verify the correctness of our proof. With limited time and resource, we only conduct experiment up to 100 nodes. We choose $\rho = 1.5$ for spatial gossip, $q = 2^8$ for alge-gossip and NBgossip.

It can be observed in Figure 3.3 that NBgossip and alge-gossip require much fewer number of rounds than uniform and spatial gossip. This could be intuitively explained as follows. In uniform and spatial gossip, the transmitted message is selected at random, the more messages a node holds, the more likely that a newly received message is a redundant one. Another problem is that towards the end of the gossip protocol, many nodes may be missing the same "rare" messages, and significant large number of rounds may be required to fill up the holes. This is quite similar to the well known coupon collector problem, in which much of the complexity results from the finishing steps. It is further observed that spatial gossip requires much more number of rounds to terminate compared with uniform gossip. As explained in Section 3.2, in spatial gossip, nodes selects

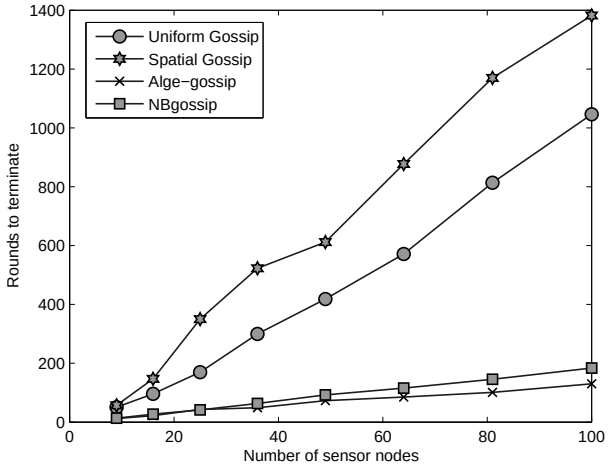

Figure 3.3: Comparison of NBgossip with uniform gossip, spatial gossip and alge-gossip in terms of rounds to terminate.

their neighboring nodes with much higher probability than those located further away. Coupled with the fact that nodes within proximity tend to share the similar message set, thus considerable large number of redundant messages are exchanged in each round. In contrast, under uniform gossip, all nodes are selected with equal chance, which alleviates the problem exhibited by spatial gossip.

On the other hand, for both NBgossip and alge-gossip, the received message is a linear combination of many messages. Even though the receiver node might have one or two messages in the combination set, it would still find the received message useful with high probability. In mathematical manner, the newly received message is likely to increase the rank of the coefficient matrix. Thus NBgossip and alge-gossip can effectively get rid of the problems associated with uniform and spatial gossip.

In addition, it is noticed that NBgossip has almost identical performance with alge-gossip but incurs slightly more rounds. This can be explained by the following argument. The probability of linear independence on a newly received message is $\prod_{s=1}^{n}(1-1/q^s)$ for alge-gossip [3], which is higher compared to $(1-1/q)^n$ in NBgossip. With $q \gg n$, it can be foreseen that the performance of NBgossip should be exactly the same as alge-gossip.

3.5.3 Comparison of the Total Energy Consumption

The total amount of energy consumption is determined by the number of rounds and energy incurred per round. Since the number of rounds and energy incurred per round are determined solely based on how the messages are exchanged between nodes. That's why one of the most important design considerations of gossip algorithms lies in the

message exchange scheme, i.e., which node to contact, and the message content to be exchanged. Once the message exchange method is fixed, the energy consumption per round and number of rounds are automatically determined. We define the energy cost for communication between neighboring nodes as 1 (unit energy consumption). The ideal energy consumption model assumes that the communication cost is symmetric and the energy cost for multi-hop communication is equal to the number of hops in between. We agree that there is additional packet overhead (due to coefficient vector) for alge-gossip and NBgossip, but the additional cost can be ignored with the following justification: The message length can easily fall in the range of kilobytes or even megabytes, which is much larger than the coefficient vector and thus renders the coefficient vector length negligible.

For spatial gossip, though energy could be reduced by picking up nearby nodes with a higher probability, the large number of rounds dominates the energy consumption. It is even worse for uniform gossip, where short and long distance communications are equally likely and multi-hop routing is known to be expensive in sensor networks. Coupled with the considerably large number of rounds needed, uniform gossip incurs the highest energy consumption as depicted in Figure 3.4. Nevertheless, to be fair, uniform gossip is originally designed for networks without energy constraints. Thus it is not surprising to see that NBgossip saves energy by 86% over spatial gossip on average and more than 90% for uniform gossip.

Being able to terminate in the optimal $O(n)$ rounds with high probability and by communicating with one-hop nodes only, NBgossip consumes the least amount of energy. While in alge-gossip, long and short distance communications are randomly determined, thus alge-gossip consumes much more energy as compared to NBgossip on a per round basis. Therefore, even though NBgossip and alge-gossip take similar number of rounds to terminate, NBgossip saves 60% energy over alge-gossip on average. The relative higher energy consumption associated with alge-gossip can be regarded as the cost of multi-hop routing when nodes located further away are selected.

3.5.4 Comparison of Computational Complexity

Besides comparing the practical performances of the four gossip protocols, the computational complexities are also evaluated. For uniform and spatial gossip, each node randomly selects a message from its message pool. Regardless of the number of rounds passed, the computational complexity remains the same. However, under alge-gossip constraints, nodes randomly combine all the messages they have received so far. At the beginning, each node owns exactly one message. As gossip process continues, the number of messages at a node increases linearly with the number of rounds. Thus after n rounds

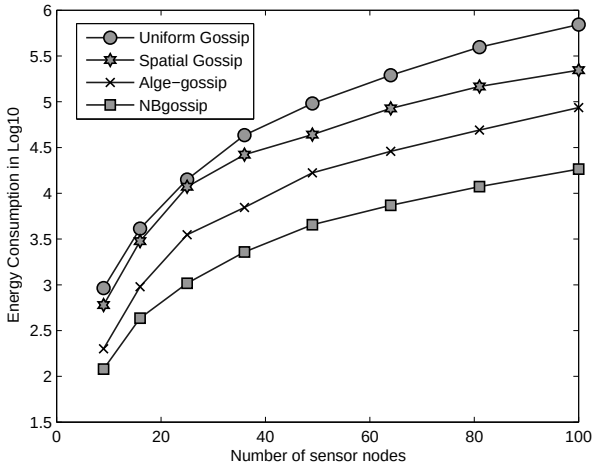

Figure 3.4: Comparison of NBgossip with uniform gossip, spatial gossip and alge-gossip in terms of energy consumption.

have passed, the linear combination would involve n messages, which results in a computational complexity of $O(n)$. Finally, in NBgossip, each node randomly combines all the messages received from its neighbors in each and every round. Since the number of neighboring nodes k is in the order of $\Theta(\log n)$, the operation required would be $\Theta(\log n)$ under all cases regardless of the number of rounds passed. In conclusion, Table 3.2 summarizes the computational complexities for NBgossip, uniform gossip, spatial gossip, and alge-gossip in worst, best, and average case, respectively.

Table 3.2: Comparison of NBgossip with uniform gossip, spatial gossip and alge-gossip in terms of computational complexity

Protocols	**Best Case**	**Worst Case**	**Average Case**
NBgossip	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$
Uniform Gossip	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
Spatial Gossip	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
Alge-gossip	1	n	$\frac{1}{2}n(O(n))$

In summary, by combining the advantages of network coding and neighborhood gossip, NBgossip outperforms all other gossip protocols in terms of energy incurred and is identical with alge-gossip in terms of average computation time (number of rounds). In addition, a $\Theta(\log n)$ computational complexity is considered scalable for wireless ad-hoc and sensor networks. In the next section, we shall comment on the real implementation issues for NBgossip in wireless ad-hoc and sensor networks.

3.6 Implementation Issues in ad-hoc and Sensor Networks

In this section, we illustrate how NBgossip can be applied in ad-hoc and sensor networks. It is known that wireless networks are inherently broadcasting in nature. Since in NBgossip, each node only communicates with its neighboring nodes, we can extend NBgossip to make use of this broadcast advantage. For node u_i, instead of sending a different message to each neighbor, it sends the same coded message to all neighbors in a single transmission. i.e.,

$$e = \sum_{x=1}^{k} \beta_x \cdot e_{xi}^S \tag{3.12}$$

This adaptation can also be viewed as a network coding combined with communication via flooding. In flooding, duplicated messages may be forwarded blindly. However, by making use of network coding, we ensure that any message received by a node is meaningful in the sense that it can increase the coefficient matrix rank. Let us denote this new NBgossip protocol as *NBbroadcast* in order to differentiate. NBbroadcast has almost identical number of super rounds to terminate as compared to NBgossip. In NBbroadcast, a super round takes up one round only, while k rounds are needed for NBgossip. As a result, NBbroadcast is faster and can further reduce the energy consumption compared to NBgossip. As shown in Figure 3.5, the energy consumption of NBbroadcast is at least 90% lower than uniform gossip, spatial gossip and alge-gossip.

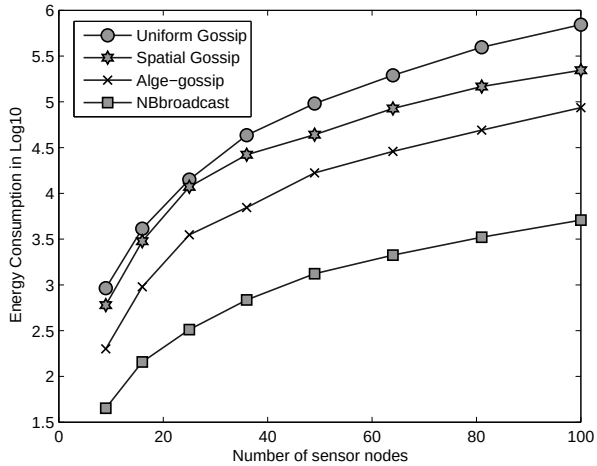

Figure 3.5: Comparison of NBbroadcast with uniform gossip, spatial gossip and alge-gossip in terms of energy consumption.

Well-known examples of existing sensor nodes are the Xbow MICA series [19]. Normally these sensor nodes have a 64KB to 128KB on-chip program flash memory and

512KB serial flash memory for data. Besides, as discussed in previous sections, all arithmetic operations are defined over finite field F_q, where $q = 2^x, x \in Z^+$, and can be realized by bit-wise operation with bit length x [12]. Moreover, the linear equations in network coding can be efficiently solved by Gaussian Elimination [17], and previous work [18] has successfully shown the possibility of implementing network coding in sensor networks. In conclusion, all these render the feasibility of applying NBgossip algorithm in wireless ad-hoc and sensor networks.

3.7 Summary

This Chapter has introduced a novel and practical solution called NBgossip, which significantly reduces the time and energy consumption required to disseminate all information in energy constrained networks. By combining the advantages of network coding and neighborhood gossip, we have proved that NBgossip could terminate in $O(n)$ rounds with high probability.

In Section 3.2, we examined three existing protocols in details and described our proposed NBgossip protocol. In the subsequent section, a formal mathematical proof on the execution time (round) for NBgossip is given and the upper and lower bounds to disseminate all information are derived. In simulation part, we implemented all the four algorithms and evaluated the performance of NBgossip, uniform gossip, spatial gossip and alge-gossip. As we have explained throughout this Chapter, though uniform and spatial gossip are easy to implement, both suffer from the "rare message" problem, and large number of rounds are wasted by passing redundant information over the network. Alge-gossip successfully gets rid of the "rare message" problem, however, it has its own drawbacks. On one hand, multi-hop routing to a particular node, which is expensive in sensor networks, is unavoidable in alge-gossip. On the other hand, alge-gossip exhibits an average $O(n)$ computational complexity, which is impractical to be implemented directly in wireless sensor networks. By combining network coding and neighborhood gossip, the proposed NBgossip eliminates the limitations associated with previous approaches. Same as alge-gossip, applying network coding helps to avoid the "rare message" problem. Meanwhile, multi-hop routing is unnecessary as communications take place among neighboring nodes only. Finally, instead of combining messages received so far, each node randomly combines messages received from neighbors in the previous super round. Consequently, the computational complexity required is in the order of $\Theta(\log n)$.

To sum up, it is found out that our NBgossip significantly reduces energy consumption as compared to uniform gossip (over 90%), spatial gossip (average 86%) and alge-gossip (average 60%). In addition, NBgossip is identical to alge-gossip in terms of number of rounds to spread all the messages. Towards the end, we commented on how NPgossip can

be applied to wireless ad-hoc and sensor networks. By extending the original NBgossip to NBbroadcast, which makes use of the wireless broadcast advantage, the time and energy incurred is further reduced. Finally, we discussed the implementation feasibility on stargate and sensor class nodes. The results of our work lead to the publication of an extended paper at IEEE Conference on Mobile Ad-hoc and Sensor Systems (MASS) and an invited paper submission to Journal of Computer Science and Technology (Springer).

Chapter 4

Closest Point Search for Computing Average

In-network data aggregation, which reduces data redundancy and hence certain network load, has been put forward as an essential paradigm for wireless ad hoc and sensor networks. Due to simplicity and robustness, gossip based algorithms for data aggregation have recently received significant attention. However, a common drawback of many gossip based protocols is the waste of energy in passing around redundant information multiple times. Thus gossip algorithms need to be re-designed in order to be applicable for energy constraint networks. In this Chapter, we study the averaging problem under the gossip constraint. In a network of n nodes, each node u_i holds a value x_i at the beginning and the objective is to compute the global average of these values in a distributed manner, while consuming the least amount of energy. By viewing the gossip process as constructing random binary data fusion trees, and formulating the fusion operation as closest point search in a n-dimensional cube, we prove that the true average can be computed in the optimal $O(\log n)$ rounds without any probability involved. Moreover, the proposed algorithm is shown to outperform existing approaches for wireless sensor network in terms of the number of radio transmissions.

4.1 Related Work

The advent of sensor and communication technology enables the construction of small, smart, and cheap computing devices known as wireless sensors. Large scale wireless sensor networks, which normally consist of hundreds or thousands of sensor nodes, offer numerous opportunities for a wide range of applications. Nodes in such networks usually operate under limited communication, computation, and energy resources. In addition, nodes may join or leave the network autonomously and no centralized entity for communication, time synchronization, and coordinating computation is available. These unique

characteristics of wireless sensor networks have triggered the design of robust, simple, scalable, and energy efficient distributed algorithms. Consider a network of n nodes, in which each node u_i has a piece of measurement data x_i, and all nodes are required to compute the average of the n sensor measurements. This problem is formally known as *averaging problem* [31] and is of particular interest to many applications, such as data redundancy reduction, sensor calibration, and distributed data fusion.

With a basic understanding of gossip protocols from previous Chapters, it is time to discuss how gossip algorithms might be used to solve the averaging problem. In every round, each node randomly picks one of its one-hop neighbors, and the two nodes exchange their current values with each other. This pair of nodes compute the pairwise average, which then becomes the new value for both nodes. By iterating this averaging process, it is shown that eventually the values at all the nodes converge to the global average [24].

The key issue in applying gossip protocols is how many rounds (iterations) needed for the local values to converge to global average with sufficient accuracy. The averaging time of a gossip algorithm turns out to be closely related to the mixing time of the Markov chain defined by a weighted random walk on the graph. Boyd *et al.* [18, 32] found that the convergence speed depends on the second largest eigenvalue of a doublely stochastic matrix and proposed a subgradient method to optimize the neighbor selection probabilities for each node in order to find the fastest mixing Markov chain on the graph. This method is proved to work on complete graphs, expander graphs, and peer-to-peer networks. However, as noted in [18, 23], for the graphs corresponding to typical wireless sensor networks, this approach can result in very high energy consumption. For example, if geometric random graph [16, 33] is assumed for sensor network topology, this gossip algorithm requires $\Theta(n^2)$ transmissions. To mitigate this, [23] proposed geographic gossip, which makes use of the geographic information of a sensor network to reduce radio transmissions. In addition, [34, 35] have also considered the problem of computing averages in networks. Even though the results are promising, their algorithms are based on the assumption that the underlying networks are regular graphs.

In this Chapter, unlike the existing approaches, where each node blindly combines values with its communication partner, we propose a new gossip protocol (*CPSgossip*), which carefully selects the message to be transmitted and randomly chooses a gossip partner. In each round, every node (caller node) informs its corresponding communication partner (called node) about the message it desires the most, and the called node replies with the most similar message it can construct. By formulating the message construction process as a closest point search problem, we can show that the nodes can compute the global average in the theoretically optimal $O(\log n)$ [18] rounds. In addition, we utilize the geographic information available in the network and apply geographic routing to gossip with random nodes located far away in the network. It is demonstrated in Section 4.5 that the extra cost associated with multi-hop routing can be compensated by the rapid diffusion of information.

4.2 Problem Formulation and Technical Approach

Each node has a unique identifier u_i, where $i \in \{1, \ldots, n\}$, and is randomly placed in an operation field. All nodes are collectively denoted as V. For ease of description, the system is assumed to operate under synchronous time model. In this model, time is divided into slots, and in each time slot, nodes contact one of their peer nodes independently. Conceptually, all nodes communicate simultaneously, but in reality, nodes within transmission radius of each other are not allowed to transmit at the same time due to interference. It is noticed that some work, for example, [23] considers asynchronous model. However, the qualitative and quantitative conclusions are unaffected by the choice of time model in analyzing gossip based algorithms [6]. Here we choose synchronous time model for convenience. The system advances in rounds indexed by $t \in \mathbf{Z}^+$, where each round occupies one time slot. In one particular round, every node u_i randomly chooses a point g in the operation filed and exchanges messages with node $u_j \in V$, $j \neq i$, which is the nearest node to g, to compute the global average. With respect to the flow of information, the push and pull transmission models are described in Chapter 2. In push model, node u_i sends a message to u_j. While for pull model, node u_j sends a message to u_i on receiving the request from u_i. Our approach makes use of the pull transmission model.

At the beginning, each node $u_i \in V$ has a measurement data x_i^0, and the ultimate goal is for all nodes to compute the global average $x_{ave} = \frac{1}{n}\sum_{i=1}^{n} x_i^0$. For the purpose of analysis, we rewrite the objective statement in a vector-based fashion. In each round, each node u_i receives a $1 \times n$ *average mark vector* v_i^t and a corresponding value x_i^t from its communication partner. The average mark vector together with the corresponding value is denoted as a *vector-value pair*. The use of the average mark vector can be explained by the following example. If $n = 5$ and the vector-value pair at a certain node u_i is $v_i^t = (11001)$ and x_i^t, then x_i^t is the average value of x_1^0, x_2^0 and x_5^0. In order to compute the average, each node must be able to construct the $1 \times n$ all-one vector. At the beginning, each node u_i has v_i^0 set to $\mathbf{e_i}$ (the vector with 1 in the i^{th} position, and 0 in all others) and x_i^0 as the measurement data. In addition, we define the *best average vector* generated by node u_i after t rounds have passed as v_i^{tb}:

$$v_i^{tb} = \arg\min_{v_i^{tb}} \left\| v_i^{tb} - \mathbf{1} \right\| \tag{4.1}$$

subject to:

$$v_i^{tb} = \sum_{k=1}^{t} \alpha_k . v_i^k \tag{4.2}$$

$$v_i^{tb} \in \{0, 1\}^n \tag{4.3}$$

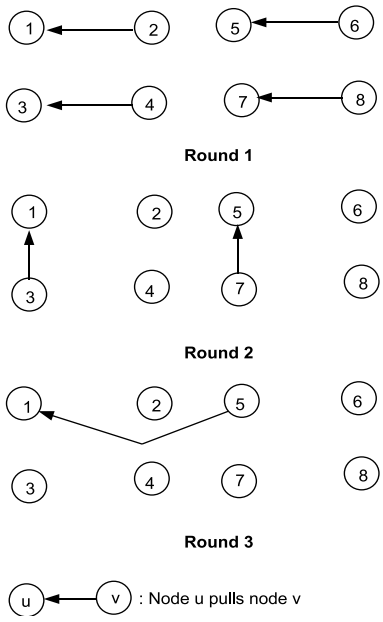

Figure 4.1: Computing global average in a 8-node network.

where $\|\cdot\|$ is the square of the Euclidean norm, α is any real number, $\mathbf{1}$ is the $1 \times n$ all-one vector and $\{0,1\}^n$ is a $1 \times n$ vector whose entries can only be 0 or 1.

Our approach stems from the following observation: for a particular round, if the average mark vector received by u_i from its partner happens to complement the best average vector it has, i.e., the received vector has alternate 0 and 1 positions compared to best average vector, then the global average can be obtained. This complement is defined as the *inverse* of the best average vector. For example, vector $v = (0110)$ is the inverse of vector (1001). If in each round, the average mark vector received from the partner has at least some 1s in the 0s positions of the best average vector and 0s in the 1s positions, the global average can be obtained in $O(\log n)$ rounds. Consider a 8-node network, as shown in Figure 4.1, node u_1 is able to compute the global average after only 3 rounds. After round 1, u_1 is able to compute the average of x_1 and x_2. At the end of round 2, u_1 gets the average of x_1, x_2, x_3 and x_4. Finally u_1 obtains the global average after 3 rounds. All communication links shown in each round must be satisfied for u_1 to get the average in 3 rounds.

However, in reality, since the operations at each node are fully distributed and no global information is available, each node has no idea about which partner to contact. Instead of blindly repeating the pairwise averaging process, we ask the nodes to tell their communication partner what they want. Under this greedy approach, each node constructs its best average vector first. After that, it sends the inverse of the best average

vector to a randomly picked communication partner. Upon receiving the pull request, the called node will generate a most similar vector (formally defined in Section 4.3) and computes the associated value accordingly. This most similar vector together with the associated value are transmitted back to the caller node. The following example explains why this is a greedy approach. Let $n = 5$ and node u_i has two average mark vectors, (11100) and (01111). Meanwhile, node u_j also has two average mark vectors (00011) and (11000). Following the above description, u_i would send the request message (10000) to u_j , and u_j would reply with message (11000). By receiving this message, it is still impossible for u_i to compute the global average. On the other hand, if (00011) were received, u_i could immediately work out the global average.

4.3 CPSgossip

This section presents a gossip protocol, whose time complexity is asymptotically optimal. The protocol employs a pull based strategy, with closest point search as its core component. In this section, the CPSgossip protocol is introduced first followed by our solution for the closest point search problem. Finally, a detailed analysis of CPSgossip algorithm is given.

4.3.1 CPSgossip Protocol

In every round t, each node u_i randomly selects a location g in the operation field and constructs a request message, which contains the complement (i.e., inversed) of best average vector, $\overline{v_i^{tb}}$, and its own location l_{u_i}. If u_i itself happens to be the closest one to g, it will select another location. Otherwise, u_i sends the request message to its one-hop neighbor that is closest to g. By iterating this forwarding process, eventually u_j, which is the closest node to g, is reached, i.e., u_j has no one-hop neighbors with a smaller distance to g. Node u_j looks into the pool of average mark vectors it has received so far, and constructs a best match vector of $\overline{v_i^{tb}}$ and sends back to u_i. To summarize, the pseudocode is presented in Algorithm (Protocol) 2. It should be noted that the locations of one-hop neighbors are usually kept by each node.

As shown in Algorithm 2, besides the best match vector v_{ji}^{t+1}, the corresponding value x_{ji}^{t+1} is also required. Based on the actual meaning of these average mark vectors, the arithmetic operation among these vectors can be defined as following:

For vector-value pair (v_1, x_1) and (v_2, x_2), if $v_3{=}\alpha_1 \cdot v_1 + \alpha_2 \cdot v_2$, then x_3 is calculated as:

$$x_3 = \frac{\alpha_1 \cdot x_1 \cdot sum(v_1) + \alpha_2 \cdot x_2 \cdot sum(v_2)}{sum(v_3)} \tag{4.4}$$

Protocol 2 CPSgossip for Computing Global Average

CompletedNodes[1 : n]=0;
while sum(CompletedNodes) $\neq n$ **do**
  System advances from round t to $t+1$
  for sensor node i=1 : n **do**
    if CompletedNodes[i] == 0 **then**
      u_i randomly picks up a point g in the operation field.
      u_i constructs the best average vector v_i^{tb} as:
      $v_i^{tb} = \arg\min_{v_i^{tb}} \left\| v_i^{tb} - \mathbf{1} \right\|$
      subject to :
      $v_i^{tb} = \sum_{k=0}^{t} \alpha_k . v_i^k$ and $v_i^{tb} \in \{0,1\}^n$.
      if v_i^{tb}==**1 then**
        CompletedNodes[i] = 1.
      else
        u_i sends out the request message.
        u_j is finally reached with the pull request message $(\overline{v_i^{tb}}, l_{u_i})$.
        u_j constructs the best match vector v_{ji}^{t+1} as:
        $v_{ji}^{t+1} = \arg\min_{v_{ji}^{t+1}} \left\| v_{ji}^{t+1} - \overline{v_i^{tb}} \right\|$
        subject to:
        $v_{ji}^{t+1} = \sum_{k=0}^{t} \alpha_k . v_j^k$ and $v_{ji}^{t+1} \in \{0,1\}^n$.
        The actual value x_{ji}^{t+1} is computed accordingly.
        u_j sends back v_{ji}^{t+1} and x_{ji}^{t+1} to u_i based on the location information l_{u_i}.
        u_i updates: v_i^{t+1}=v_{ji}^{t+1}, x_i^{t+1}=x_{ji}^{t+1}.
      end if
    end if
  end for
end while

$$sum(v) = \sum_{k=1}^{n} v(k) \tag{4.5}$$

where $v(k)$ is the value at the k^{th} position of v.

Meanwhile, if a node u_j has already obtained the all-one vector-value pair $(\mathbf{1}, x_{ave})$, then regardless of what u_i requests, it will transmit the all-one vector-value pair. Moreover, in [4], Karp *et al.* have shown that a new message needs $O(\log n)$ rounds to be disseminated to all nodes. Thus, if any of the nodes obtains the all-one vector-value pair, the next $O(\log n)$ rounds would ensure that this average value is spread to all nodes.

4.3.2 Closest Point Search

It is intuitive to see that the core part of the proposed gossip algorithm is to construct the best average vector or best match vector from a set of known vectors. This problem is more commonly known as *Closest Point Search (CPS)* [36, 37] or *Closest Vector Problem (CVP)* [38]. In CPS or CVP, a generator matrix G is given with real entries whose rows are linearly independent over $\mathbf{R}$, and a lattice is generated based on G. Let m and n denote the number of rows and columns in G respectively, the lattice can be represented as:

$$\Lambda(G) = \{u \cdot G : u \in \mathbf{Z^m}\} \tag{4.6}$$

The CPS or CVP is the problem of finding, for a given Λ and a given point $x \in \mathbf{R^n}$, a vector $\overline{c} \in \Lambda$ such that:

$$\|x - \overline{c}\| \leq \|x - c\|, \forall c \in \Lambda \tag{4.7}$$

For a general CPS problem where there is no structure in the lattice generated, finding the closest point is NP-hard [13, 14]. However, the choice of method for solving this closest point search problem is shown to be dependent on the structure of the lattice [36]. Intuitively, the more structured a lattice is, the faster can the closest point be found. The generator matrix G for our problem is made up of the average mark vectors received so far, i.e., $G_i = (v_i^0, v_i^1, \ldots, v_i^t)^T$. The lattice generated is:

$$\Lambda(G_i) = \{v | v = u \cdot G_i, u \in \mathbf{R^m}\} \tag{4.8}$$

and

$$v \in \{0, 1\}^n \tag{4.9}$$

For a given point $x \in \{0, 1\}^n$, the objective is to find a vector $\overline{v} \in \Lambda(G_i)$ such that:

$$\|x - \overline{v}\| \leq \|x - v\|, \forall v \in \Lambda(G_i) \tag{4.10}$$

A common approach to the closest point search problem is to identify a certain region in the lattices which contains the optimal point. Once the region is fixed, the points within

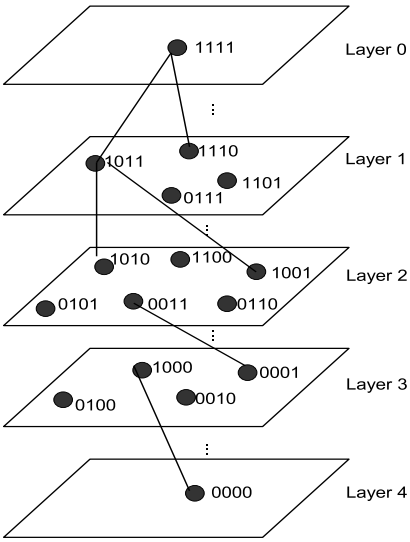

Figure 4.2: The layered structure of a 4-dimensional cube. Please note that not all edges are shown.

the region are investigated, possibly reducing the size of the region dynamically. In our case, the lattice points correspond to a subset of a $n-$dimensional cube which lies in the subspace spanned by the average mark vectors received so far. However, unlike the usual representation of a n-dimensional cube, we break down the the n-dimensional cube into $(n+1)$ layers. The vectors on the l^{th} layer have l bits difference from the objective vectors, where $0 \leq l \leq n$. In addition, two vectors are connected to each other if they have 1-bit difference. An example of such representation for $n = 4$ and objective vector (1111) is shown in Figure 4.2.

If the lattice points are not required to be on the n-dimensional cube, the closest point can be found by making use of linear least squares method. Put in mathematical form, the closest point $\hat{v}$ can be regarded as the result of the approximate solution for (4.11).

$$G_i^T \cdot \hat{\mathbf{y}} \approx x^T \tag{4.11}$$

$$\hat{v} = (G_i^T \cdot \hat{\mathbf{y}})^T \tag{4.12}$$

Applying vector calculus, $\hat{\mathbf{y}}$ can be obtained by $\hat{\mathbf{y}} = (G_i^T G_i) G_i^T x^T$. Even though the closest point obtained in this manner is not the desired one, it helps us identify the regions to which the true closest point belongs, as explained below. For any vectors in the subspace spanned by G_i, $\hat{v}$ has the minimal square distance to the objective vector, and $\hat{v}$ is actually the projection of x in this subspace. Let $d_\perp = \|x - \hat{v}\|$ and $v_\perp = x - \hat{v}$, for any vector v in this subspace, we have:

$$v_\perp \cdot v^T = 0 \tag{4.13}$$

$$\|x - v\| = d_\perp + \|v - \hat{v}\| \tag{4.14}$$

Please note that for $v \in \{0,1\}^n$, the squared norm-2 distance $\|x-v\|$ is exactly the same as the number of different bits between two vectors, i.e., $\|x - v\| = x \oplus v$. Due to the structure of the lattice, in reality, $\hat{v}$ is digitalized with 0.5 as the threshold, i.e., if the value at the i^{th} position of $\hat{v}$ is greater than 0.5, it is set to 1, otherwise it is set to 0. Let d_e denote the squared norm-2 distance between the original $\hat{v}$ and its binary version. For the closest vector $\overline{v}$ to be found in layer l, it must satisfy the following conditions:

$$\overline{v} \oplus x = l \tag{4.15}$$

$$Rank(G_i^T) = Rank(G_i^T; \overline{v}) \tag{4.16}$$

$$\lceil l - d_\perp - d_e \rceil \leq \overline{v} \oplus \hat{v} \leq \lfloor l - d_\perp + d_e \rfloor \tag{4.17}$$

The condition (4.17) effectively restricts the number of vectors to search in each layer from C_n^l to $C_{n-d_c}^{l-d_c}$, where d_c is the number of bit differences between the objective vector and the binarilized $\hat{v}$. In addition, we only need to search from layer d_c all the way to layer s, where:

$$s = \min(x \oplus v_i^k), 0 \leq k \leq t \tag{4.18}$$

In order to optimize the performance, another criteria is enforced. The value at the i^{th} position of $\overline{v}$ can not be 1 if the i^{th} column of G_i is all-zero. The first vector that satisfies the above conditions will be the desired $\overline{v}$. The following phenomenon is observed during execution. When $d_\perp \geq n/2$, it is likely that $\overline{v} = \hat{v}$. On the other hand, when $d_\perp \leq n/2$, only the first few layers need to be searched to find out the $\overline{v}$.

4.3.3 Protocol Analysis

In order to facilitate the analysis of CPSgossip, a time sliced communication graph is introduced. For example, Figure 4.3 is the result of mapping Figure 4.1 to its corresponding communication graph. From bottom to top, the edges between two layers represent the result of caller node receiving message from called node.

Assuming t rounds have passed, to perform the mapping, we first generate $t+1$ copies of the set of nodes V, denoted by $V_0, V_1, \ldots, V_t$. For convenience, we label the set of nodes at start as V_0, and arrange $V_0, V_1, \ldots, V_t$ in a vertical manner as shown in Figure 4.3. In addition, u_i^t is used to denote node u_i's clone at layer t. If the random node called by node u_i is u_j in round t, then there will be an edge connecting u_i^t at layer V_t and u_j^{t-1} at layer V_{t-1}. Finally, each node must have an edge connected back to itself from layer V_t to layer V_{t-1} according to the protocol description.

The results of this transformation are n binary trees which are rooted at the n nodes at layer V_t and have all leaves residing in V_0. We refer the binary tree rooted at u_i^t as T_i.

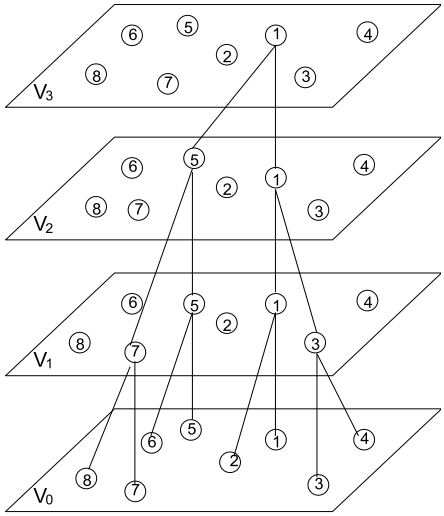

Figure 4.3: An example of time sliced communication graph. Not all edges are shown.

For ease of analysis, let us assume that the operation to merge two child nodes into their parent is a simple element-wise logical-OR of their best average vectors. Without loss of generality, we take u_1 as an example. After t rounds have passed, it is not hard to see that the number of 1s in u_1's best average vector is exactly the number of leave nodes T_1 has. Based on the tree structure, if some node u_i at layer k, $0 < k < t$, is an intermediate node of T_1, then u_i^0 must be one of the leave nodes of T_1. Since the communication partners are randomly selected, the binary trees can be constructed equivalently from top to bottom. From layer V_k to V_{k-1}, if a certain node at layer V_k belongs to the tree, there will be two edges connected to layer V_{k-1}: one edge connects to the node itself, and the other connects to a randomly selected node from V. By repeating this process, when the tree reaches height t, all together there will be $O(2^t)$ choices [22] to randomly select a node from set V to put into the tree. In order to compute the global average, we must have each node in set V selected at least once. Therefore, we need $t = O(\log n)$ rounds for a node to be capable of computing the average based on the fact that it requires $\Theta(n \log n)$ balls thrown randomly to cover n bins with high probability [16, 17]. In conclusion, the result of CPSgossip can be regarded as n random binary data fusion trees with every node as the sink node, and the whole process completes in $O(\log n)$ rounds, which is further supported by the simulation results in Section 4.4.

However, based on the understanding of average mark vectors, the ideal element-wise logical-OR operation for two nodes to aggregate their average mark vector is not realistic. Thus closest point search method is introduced to approximate this operation. This implies that all nodes need to store their previously received vector-value pairs.

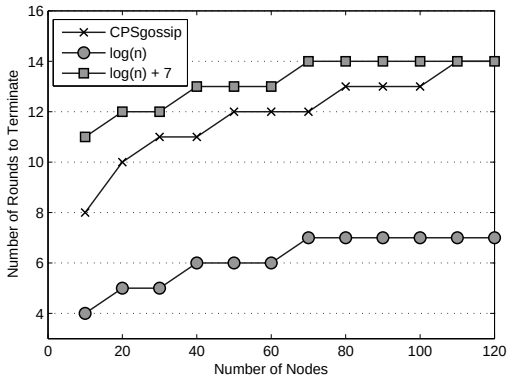

Figure 4.4: Number of rounds to computer the average with respect to the size of the network

Since the protocol terminates in $O(\log n)$ rounds, only $O(\log n)$-order memory space is required. In addition, to optimize the performance, if a called node is able to compute the global average, regardless of what the caller node requests, the all-one vector-value pair is always transmitted.

4.4 Simulation Results and Discussion

In this section, the performance of CPSgossip is evaluated in terms of the number of rounds for all nodes to compute the global average. In the network, each node has exactly one piece of measurement data to start with. Since only the number of rounds is concerned, the average mark vectors are transmitted across the network instead of the actual vector-value pair. As described in Section 4.3, each node constructs the best average vector and sends the inverse of this best average vector to its communication partner. The called node, upon receiving this request message, constructs a best match vector and sends back to the caller node.

Figure 4.4 shows the mean completion time (number of rounds) of the CPSgossip protocol, which is obtained by averaging the total number of rounds incurred over 20 executions. In addition, the simulation is conducted for network size from 10 nodes all the way to 120 nodes. As we can see, the average number of rounds to terminate is bounded by $\log n$ and $\log n + c$, where c is a constant. Here c can be interpreted as the cost associated with this greedy approach. Although the number of nodes increases linearly, the average number of rounds increases in the order of $O(\log n)$, which supports the conclusion made in Section 4.3.

Besides evaluating the system as a whole, the individual node behavior is also studied. Section 4.2 has explained the use of best average vector. The more 1s the best average

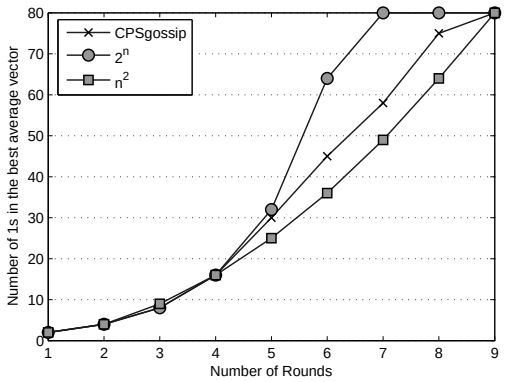

Figure 4.5: Number of 1s in the best average vector versus the number of rounds for node u_1.

vector has, the average of more nodes is obtained. Since the communication partners are randomly decided, node u_1 is selected without loss of generality. After each round, the number of 1s in the best average vector constructed by u_1 is counted. Figure 4.5 shows the number of 1s versus the number of rounds for node size $n = 80$. The values are averaged over 20 executions. It is observed that the number of 1s increases exponentially for the first few rounds, after which, the value is increased in between an exponential and square like manner. This phenomenon can be explained as follows. For the first few rounds, for any two node u_i and u_j, the probability is very low for these two nodes to select the same partner in each and every round. Thus regardless of which node is selected as the gossip partner, the average mark vectors at the two nodes are likely to be somehow complement to each other. Towards the end, when the best average vector is almost filled with 1s, due to the non-optimality of greedy approach, the increment slows down.

It is worth pointing out that if any node works out the global average, this global average will be disseminated to the network in $O(\log n)$ rounds in the worst case. This helps to avoid the situation in which the nodes are waiting for the last one or two holes to be filled with 1s. In conclusion, by formulating the message selection as a closest point search in a n-dimensional cube, the proposed CPSgossip protocol is able to compute the global average in $O(\log n)$ rounds.

4.5 Comparsion with Existing Work

In this section, we study the performance of CPSgossip in wireless sensor network and compare our results with the recent work in this field. The most commonly used network model for wireless sensor network is the one proposed by Gupta and Kumar [33]. In this

model, n sensor nodes are randomly and uniformly placed in a d-dimensional cube. Two nodes are connected to each other if they are within transmission radius r, i.e., there is a bidirectional edge between them. The established model is denoted as $G^d(n, r)$. It is well known that the transmission radius $r(n)$ has to be scaled in the order of $\Theta(\sqrt{\frac{\log n}{n}})$ [39, 40]. For simplicity of analysis, a unit square is assumed for the operation field and communications within r always succeed.

Let R^t denote the number of one-hop radio transmissions for a node u to reach the randomly selected location g in some round t, i.e., the cost of routing the message from node u to node v, and v is the nearest node to g. The same energy model , which is introduced in Chapter 3, is used here. Wireless sensor localization is already a well studied problem [41, 42], and geographic knowledge is assumed to be possessed by the network. Based on the results from the recent work on geographic routing [43, 44], the expected cost of R^t is in the order of $O(\sqrt{\frac{n}{\log n}})$. In each round, there are n such nodes and all together there are $O(\log n)$ rounds, thus the total number of message transmission C_{cps} is:

$$C_{cps} = O(n^{1.5}\sqrt{\log n}) \tag{4.19}$$

A series of work [32, 23, 18, 34, 35] have been done recently on computing averages in networks. However, [34, 35] are proved to work on regular graphs and not yet for geometric random graphs. In [18, 32], their standard gossip protocol gets the average in a total number of C_{sg} radio transmissions with probability parameter ϵ as shown below:

$$C_{sg} = \Theta(\frac{n^2}{\log n}\log \varepsilon^{-1}) \tag{4.20}$$

where the probability that the computed average is very close to the true average is $1-\epsilon$. For ϵ scaling in the order of n^{-a} for any $a > 0$, this transmission cost will be in the order of $\Theta(n^2)$. To mitigate, [23] proposed the geographic gossip, which utilizes the geographic information available in the network. The total number of radio transmissions, C_{gg} is:

$$C_{gg} = O(\frac{n^{1.5}}{\sqrt{\log n}}\log \varepsilon^{-1}) \tag{4.21}$$

Moreover, by setting $\epsilon = n^{-a}$, the final number of radio transmissions is $O(an^{1.5}\sqrt{\log n}))$. Equations (4.19) - (4.21) show that the proposed CPSgossip performs better than standard gossip and is identical with geographic gossip in asymptotic order.

In addition, CPSgossip offers another two advantages compared to standard gossip and geographic gossip. The first is that the global average computed in CPSgossip is indeed the true global average without any probability involved. The other one is that CPSgossip is not associated with any estimation accuracy parameters and always terminates with a relatively small energy consumption. In contrast, the number of radio

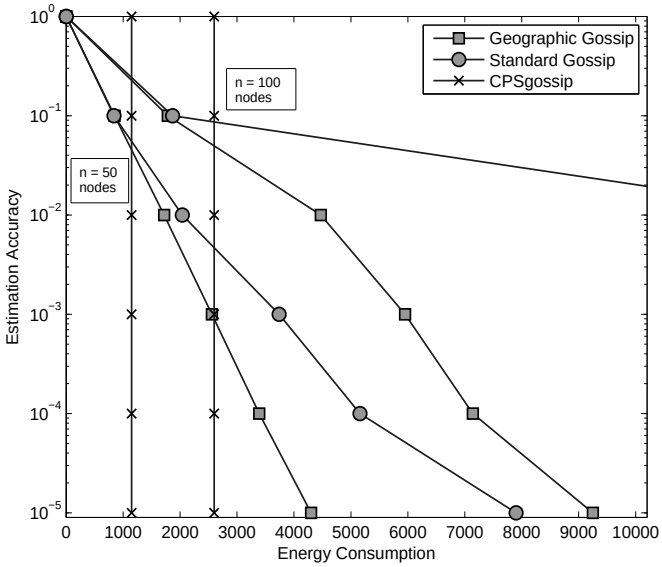

Figure 4.6: Comparison of CPSgossip, geographic gossip, and standard gossip in terms of energy consumption versus estimation accuracy to compute the average. The energy consumption is defined on number of radio transmissions and the accuracy is on log-scale.

transmissions for standard and geographic gossip in practice can be very large. Figure 4.6 depicts the energy spent versus the estimation accuracy for CPSgossip, geographic gossip, and stand gossip with node size $n = 50$ and $n = 100$. The result is obtained over a smoothly diffused temperature field as used in [7], but placing two more temperature sources. As shown in Figure 4.6, due to the absence of accuracy parameter in CPSgossip, the accuracy-energy line is a vertical one, i.e., given a network size n, CPSgossip always terminates with the true global average and a constant energy consumption. In contrast, the energy consumptions of geographic gossip and standard gossip increase at least linearly with the estimation accuracy. Which means, if the estimation accuracy increases from 10^{-1} to 10^{-2}, the number of radio transmissions at least doubles. Additionally, we can see that the intersections of accuracy-energy lines of CPSgossip and standard gossip lie around 10^{-2} and $10^{-1.5}$ for network size 50 and 100, respectively. Following this trend, one can foresee that as the network size grows, with the same energy consumption of CPSgossip, standard gossip and geographic gossip would suffer a lower and lower estimation accuracy. However, for geographic gossip, being in the same asymptotical order of CPSgossip, the estimation accuracy would eventually converge to a constant value.

4.6 Summary

In this Chapter we have proposed a novel gossip algorithm for computing averages in network in a complete deterministic and distributed manner. By formulating the message selection as a closest point search problem, the CPSgossip protocol gets rid of the uncertainty involved in many existing algorithms. Under CPSgossip protocol, nodes are allowed to communicate with any other nodes in the network. Moreover, each node asks its gossip partner for the message it desired rather than blindly combining values with partner. Towards the end, the performance of CPSgossip in wireless sensor network is evaluated and compared with existing gossip algorithms. The CPSgossip consumes significant less energy than standard gossip and is in the same asymptotic order as geographic gossip. Moreover, due to the deterministic nature, in practice, the CPSgossip requires fewer number of radio transmissions as compared to geographic gossip. Finally, the proposed protocol has no requirement on the underlying network model, and hence makes it applicable for other networks as well. This work leads to a publication in IEEE SENSORCOM and a journal submission to Elsevier Ad Hoc Networks.

Chapter 5

A Belief Propagation Approach towards Wireless Sensor Calibration

Once deployed in the field, even pre-calibrated sensors are prone to various faults that corrupt the sensor data. Numerous factors contribute to errors in sensor measurement. In a large-scale sensor network, it is impossible to calibrate the sensor device one by one manually. In addition, it is very cumbersome to detect and fix these errors in a timely fashion after deployment. Thus the sensors must be intelligent enough to perform self-calibration. In this Chapter, we present an automatic, in situ sensor calibration algorithm based on loopy belief propagation. Each sensor could have its own observation, which may or may not be the same as the true value. However, sensors are assumed to possess the true value relationships among their neighbors. Once observations are made, sensors will initiate and distribute message passing to infer the real values until the system converged to an acceptance threshold according to loopy belief propagation.

5.1 Introduction

The recent advent of sensor and communication technology enables the construction of small, smart and cheap sensing & computing devices known as wireless sensors. Large scale networks, which usually consist of hundreds or thousands of sensors, open a new paradigm for a wide range of applications; examples are tracking, environment monitoring, urban sensing, military surveillance, factory automation and many others [45]. However, due to the uniqueness of sensor networks, it also poses lots of new challenges in the field of distributed and pervasive computing. Distributed sensor calibration ensures the correctness of data, which is essential for the performance of the whole network, while at the same time. Which generally overlooked by many researchers. The measured value of a sensor often has a deviation from the true physical state. The difference between the two is commonly known as bias, which can be further decomposed into systematic error

and random noise [46]. Typically, random noise can be effectively reduced by taking the average of a number of samples. However, systematic error is a property of the sensor which depends on time, sensed phenomena, and the environment. The bias of a sensor is typically determined in the factory through calibration, which is the process of mapping raw sensor readings into corrected values by identifying and correcting systematic bias. However, even the best calibrated sensors, once deployed in the physical field, are prone to faults that corrupt the sensor data. Thus, regular calibration of the sensors is necessary for the correctness of the data. Based on the techniques used, calibration can be broadly divided into active calibration and passive calibration. Active calibration techniques rely on specific stimulus with known results and are typically implemented via actuator elements. In contrast, passive calibration techniques do not require external stimulus; instead, it exploits the spatial and temporal correlation among sensor data.

For certain wireless sensor network (WSN) applications, physical access to the deployment site of the sensors is difficult or even impossible. Thus it may not be practical to perform active calibration. Besides, active calibration is expensive as external actuator infrastructure is needed. Moreover, active calibration is usually not scalable. Therefore, in general, passive calibration is more favorable due to its distributed nature, simplicity, and low cost. In our approach, passive calibration technique is used. We model the sensor network as a pairwise Gaussian Markov Random Field (GMRF) and exploit the correlations among neighboring sensors. Once the sensor measurement data are obtained, loopy belief propagation is applied to infer the real values.

The topic of device calibration is as old as the device themselves. In this section, we review two most cited methodologies on calibration with particular reference to wireless sensor networks. In [47], the authors propose a two-phase post-deployment calibration technique for large-scale, dense sensor deployments. In the first phase, the algorithm derives relative calibration relation among pairs of co-located sensors; while in the second phase, it maximizes the consistency of the pair-wise calibration functions among groups of sensor nodes. The key idea in the first phase is to use spatial correlation of signals received at neighboring sensors when the signals are highly correlated. The underlying assumption is that the same phenomena will have identical effect on neighboring nodes. In addition, full knowledge of the correlation window is assumed. However, in reality, these two assumptions may not always be valid. In the paper written by K. Whitehouse and D. Culler [48], they addressed the problem of calibrating the transmission power in the context of signal strength with the distance between two locations. In their approach, each individual device is parameterized and the system responses are modeled as a whole. Once the distance information is collected, the parameters are solved as a global optimization problem. Since the distance information among connected nodes is required, it is regarded as an active calibration technique. Additionally, [49, 50, 51]made their own contributions. [49] applies nonparametric belief propagation to calibrate the

distance information among sensor nodes which is somewhat similar to our approach. In [50], they calibrate the randomly distributed sensor array based on tracking a single target moving at constant velocity using extended Kalman filter at each sensor. [51] uses a mobile actuator to calibrate sensor nodes.

5.2 Loopy Belief Propagation

Belief Propagation has been successfully applied to many different fields including computer vision, artificial intelligence, statistical physics, and coding theory [52]. The rationale behind belief propagation is an efficient way to solve the inference problem by propagating local messages around neighborhoods. In this thesis, we show that belief propagation and its generalization can be used as an efficient way to solve the wireless sensor calibration problem.

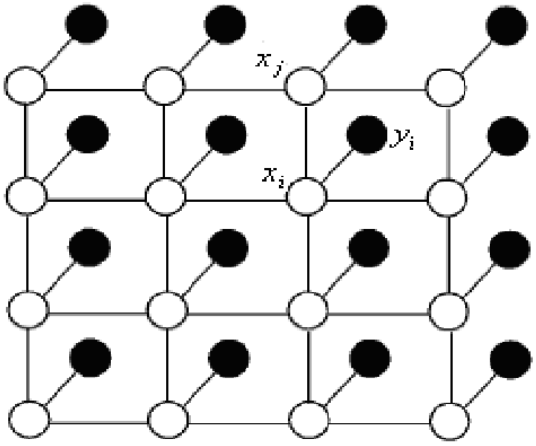

Figure 5.1: A grid layout pairwise Markov random field.

5.2.1 Pairwise Markov Random Field

In many practical problems, we are mainly interested in the pairwise relation among the variables. Normally a set of data is given and we want to infer a representation of whatever is "really out there". A graphical depiction of this model is shown in Figure 5.1. The solid circle y_i represents the sensor observed value while the empty circle x_i represents the real value in the context of sensor calibration. The index i means the i^{th} sensor node. Normally there is some statistical dependency between the observed and

real value of each sensor node. Here we denote the joint probability function (potential function) as $\phi(x_i, y_i)$. In statistics, the function $\phi(x_i, y_i)$ is often referred as "the evidence for x_i". Furthermore, due to the spatial correlation among neighbors, there has to be some structure for x_is. In general, if we do not assume such structure, passive calibration are inherently ill-posed. As depicted in Figure 5.1, the sensor nodes are organized in a two-dimensional grid manner for simplicity. For each node i, there is some statistical relationship between its own real value and its neighbor j's real value, which is denoted as $\varphi_{ij}(x_i, x_j)$. The joint probability of the graph configuration can be written as:

$$P(X, Y) = \frac{1}{Z} \prod_{(i,j)} \varphi_{ij}(x_i, x_j) \prod_i \phi_i(x_i, y_i) \tag{5.1}$$

5.2.2 Loopy Belief Propagation Inference

To solve the Markov random network, the $\varphi()$s and $\phi()$s must be known in the first place. Normally these functions are obtained from observed data and past history, or even well-established theories. The next step will be estimating the marginal posterior distribution or maximum a posterior (MAP) of x_is given the observed y_is. This is where belief propagation algorithm steps in. For acyclic graph, belief propagation is an exact estimation methods. While for graph with loops, belief propagation is not guaranteed to converge and is more commonly known as loopy belief propagation. However, loopy belief propagation has been successfully applied in a wide range of applications [52]. A brief description of loopy belief propagation is given below.

The belief propagation algorithm iteratively sends messages in the network. Messages are passed between neighboring nodes only in a local manner. The message from node i to j is represented as $m_{ij}(x_j)$, which tells how likely node i thinks that node j is in state x_j, the probability distribution function of x_j from node i's point of view. Depending on whether marginal or MAP estimator is used, two types of messages can be passed in the network. For discrete variables, the two types of messages can be written as:

Marginal probability:

$$m_{ij}(x_j)_{t+1} = \sum_{x_i} \phi_i(x_i, y_i) \varphi_{ij}(x_i, x_j) \prod_k m_{ki}(x_i)_t \tag{5.2}$$

MAP estimator:

$$m_{ij}(x_j)_{t+1} = \max_{x_i} \phi_i(x_i, y_i) \varphi_{ij}(x_i, x_j) \prod_k m_{ki}(x_i)_t \tag{5.3}$$

where $k \in N(i), k \neq i$ and $m_{ki}(x_i)_t$ is the message computed in the last iteration of belief propagation.

When messages converge, the final belief at each node $b(x_i)$ can be obtained. Again there are two ways to compute the local belief:

Marginal probability:

$$b(x_i) = k\phi(x_i, y_i) \prod_j m_{ji}(x_i) \tag{5.4}$$

where k is the normalization constant.

MAP estimator:

$$b(x_i) = \arg\max_{x_i} \phi(x_i, y_i) \prod_j m_{ji}(x_i) \tag{5.5}$$

where j is the neighboring nodes of i.

5.3 Effective Sensor Calibration

Our approach here towards wireless sensor calibration is to model the sensor network as a pairwise MRF. Each sensor has its own measurement. These measurements are assumed to obey some probability function with their real physical value respectively. For neighboring nodes, another set of probability distributions govern the relations between real physical values among them. After the measurements are made, loopy belief propagation is applied to infer the real physical values. For the rest of this section, we will choose light sensors as an example and illustrate how the loopy belief propagation is applied.

Figure 5.2: The projector surface and sensor grids.

5.3.1 Problem Description

It has been observed that the light intensity values returned by two light sensors are significantly different even if they are placed side by side under the same lighting environment. This motivates our research on the sensor calibration. In order to control the lighting environment, the wireless sensors are put in a room where the only light source is a projector. The projection surface is equally divided into a 4x4 grid and the sensors are randomly placed in each grid (refer to Figure 5.2). The projector projects a particular light intensity value on each grid based on some probability distribution. As the distances from the grids to the projector are different, even if a same intensity value is projected to all grids, a sensor node may still have different observed values across different grids. Equivalently, we can predict that when a sensor is sequentially put in 2 different grids, to which we assign carefully designed light intensities, it is highly possible that the sensor returns the same observed value even if the real values are different. Therefore, when sensors are randomly placed in the projection surface, given the observed value of a sensor, there might be a few possible real intensity values projected to the grid the sensor is placed on. Intuitively this gives us the potential function between observed and real value. The potential functions for the intensity values between adjacent grids are pre-defined to simulate the real world light intensity distribution. As a result, the following probability functions are obtained:

$$\phi(x_i, y_i) = p(x_i|y_i) \tag{5.6}$$

$$\varphi(x_i, x_j) = p(x_i, x_j) \tag{5.7}$$

5.3.2 Formulation of Potential Functions

The potential function between x_i and x_j is defined as a mixture of bivariate Gaussian distribution with co-variance of 0.8 and variance of 200. The mean vector for each Gaussian distribution and the weight factor associated are shown in Table 5.1.

The Gaussian mixture model is a good approximation for complex probability distribution function. This explains why we choose it to simulate the potential function between adjacent nodes. Furthermore, we discretize the potential function and limit its range from 225 to 255 with step size one. Figure 5.3 shows the joint probability graphically. In this Chapter we use the same $\varphi(x_i, x_j)$ for all neighboring grids. The left horizontal axis represents the intensity value of x_i, the right horizontal axis indicates the intensity value of x_j, and the vertical axis tells the joint probability of x_i and x_j.

Once we have the potential functions among adjacent grids, the sensor nodes are placed in each and every grid, and the observation data for all possible intensities are

Table 5.1: Bivariate Gaussian mean and weight factor

$\bar{x_i}$	$\bar{x_j}$	**Weight**
230	230	0.0875
230	240	0.1
230	250	0.0875
240	230	0.1
240	240	0.15
240	250	0.15
250	230	0.0875
250	240	0.15
250	250	0.0875

recorded down. For example, we place sensor node 3 at grid 1, set the grid intensity to $255, 256, \ldots, 255$, and record down 10 sensor observed value for each of the intensities. After which, we repeat the same procedure for all grids. Figure 5.4 displays the relationship between the observed value and real value of light intensity for node 4. Please note that the observed values are the raw sensor reading without any attempt to convert to intensity values. The left horizontal axis corresponds to observed values while the right horizontal axis shows the grid light intensity. It is clear from the above figure that for a given sensor observation value, there are many possible intensity levels.

5.3.3 Inference of Real Values

Let us denote the real intensity value at projection surface grid i as x_i, the measurement data at grid i as y_i. In order to assess the effectiveness of the loopy belief propagation approach, three sets of data are needed: the real intensity values at each grid, the most likely (ML) intensity at each grid based on $\phi(x_i, y_i) = p(x_i|y_i)$ from the observed data, and the calibrated value after loopy belief propagation is applied. The real values for the grids can be seen as a 16x1 data sample drawn from all the potential function among grids. We use Gibbs sampling to achieve this with looping length equal to 100. Ideally, the distance between real value and calibrated value should be less than the distance between real value and ML value. After we have studied the problem thoroughly, the next step is to set up the environment and conduct experiments to verify our claims.

5.4 Results and Discussion

In this section, we will explain the experiment set up in details first, followed by presenting the results and discussion.

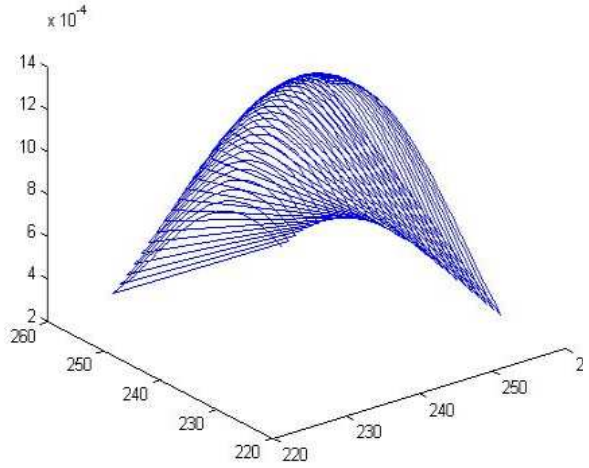

Figure 5.3: The 3D plot of neighboring potential function. Please refer to the main text for axis unit.

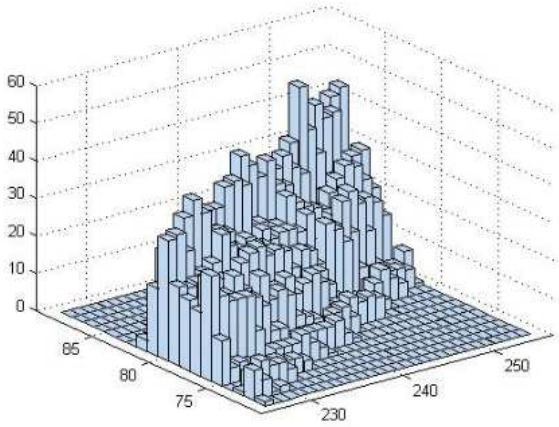

Figure 5.4: The histogram between observed and real values. Please refer to the main text for axis unit.

In this experiment, we use a 3M projector as the sole light source in a room. The sensors are randomly placed on the projection surface grids. Each sensor node comprise the Xbow MicaZ processor board and Xbow MTS310/MTS300 sensor board [53]. The

sensor data are transmitted back for processing through a gateway. The reason why we do not run belief propagation on sensor nodes themselves is that we are still in the stage of verifying the correctness of loopy belief propagation approach. However, we purposely make the potential function discrete, which can be easily ported to sensor nodes once the approach is proved to be correct. Figure 5.5 shows the detail steps involved in one cycle.

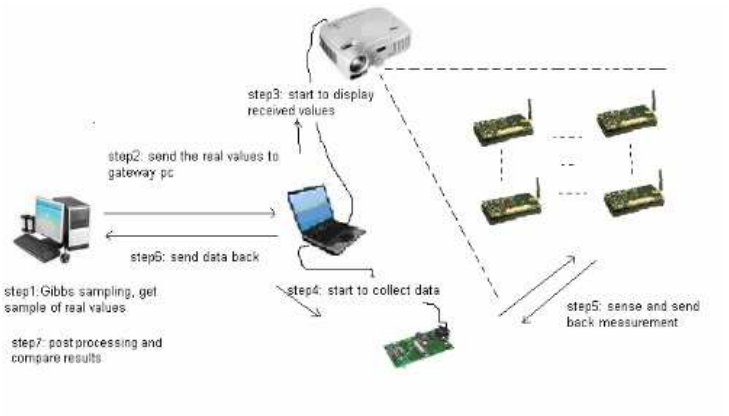

Figure 5.5: A cycle of measurement collection and calibration.

With the collected measurement data, the following comparisons are made. We define *ML error* of a sensor node to be the absolute difference between real value and ML value, and *Calibrated error* as the absolute difference between real value and calibrated value. Figure 5.6 shows the comparison of ML error and Calibrated error of one randomly selected cycle. In addition, for all 24 cycles, we compare the R^{16} Euclidean distance between real values and ML values, and the distance between real values and calibrated values. See Figure 5.7 for the comparison results. To facilitate analysis, the real intensity value is assumed to be known.

Please note that all data is sent back to desktop for further processing. From previous section, it is clear that the calibrated and ML values can be calculated accordingly. We expect the calibrated value to be closer to the real value compared to the ML value. This expectation is met exactly as illustrated in Figure 5.6 and Figure 5.7. In Figure 5.6, we can see that for Node 11, the ML error is 8. In contrast, after calibration, the error reduced to 1. However, it does not mean that every calibrated value should be closer to real value compared to ML value. Since the calibrated values are the MAP assignment, the maximal value is achieved in a global sense rather than local. This also explains why the Euclidean distance, which is a global property, between calibrated and real values is smaller than the Euclidean distance between ML and real values. In our experiment, the

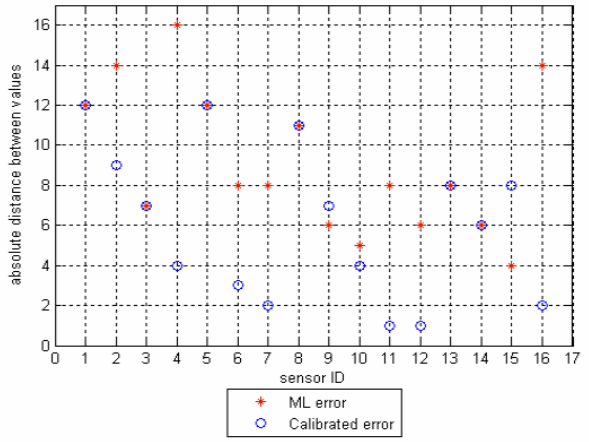

Figure 5.6: The comparison between the ML error and the Calibrated error for each of the 16 sensor nodes in one cycle. In this particular instance, calibrated error might coincident with ML error at some nodes.

Euclidean distance between calibrated and real value is on average 4.88 smaller than the distance between ML and real values. Considering that the typical Euclidean distance between ML and real values is around 30, this is indeed a significant improvement. In this experiment, only 16 sensor nodes are used. It is predicted that the performance should be even better with more sensor nodes.

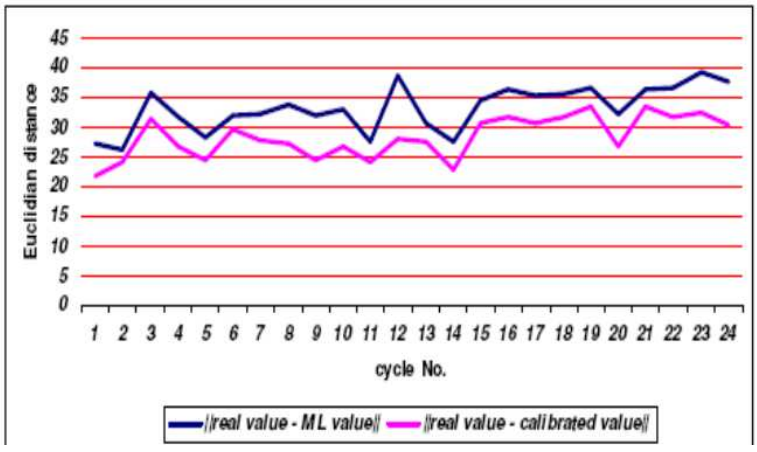

Figure 5.7: The comparison between the Euclidean distance of real values and ML values, and the distance of real values and calibrated values.

5.5 Summary

We reviewed the calibration techniques for wireless sensor nodes and propose a loopy belief propagation approach to solve the calibration problem. Throughout the Chapter, we elaborated how the experiment was set up and the potential functions obtained. In the end, the experiment results are presented, which prove the effectiveness of our approach to handle the sensor calibration problem. The key idea of our approach is to exploit the spatial correlation and apply loopy belief propagation to infer the MAP assignment of the real values. High-density systems are more fault-tolerant to node failure, provide greater opportunity to eliminate channel noise and allow nodes to sleep for longer periods and extend the lifetime of the system. Thus high density is typically desired in WSN. At the same time, high density also ensures high spatial correlation between neighboring nodes.

Chapter 6

Conclusion and Future Work

Smart environments represent the next evolutionary development step in building, utilities, industrial, home, shipboard, and transportation system automation. Like any sentient organism, the smart environment relies first and foremost on sensory data from the real world. Sensory data comes from multiple sensors of different modalities in distributed locations. The challenges in the hierarchy of: detecting the relevant quantities, monitoring and collecting the data, assessing and evaluating the information, formulating meaningful user display, and performing decision-making and alarm functions are enormous. The information needed by smart environments is provided by distributed wireless sensor networks, which are responsible for sensing as well as transporting the data. In this report, we have addressed three typical problems encountered in data-centric wireless sensor networks; namely information dissemination, data aggregation, and data calibration.

In Chapter 3, we introduced a novel and practical solution for information spreading called NBgossip, which significant reduces the time and energy consumption required in energy constraint networks. By combining the advantages provided by random network coding and neighborhood gossip, we proved that NBgossip could terminate in the optimal $O(n)$ rounds with high probability. Moreover, we compared our solution with three existing protocols in detail. It is found out that our NBgossip significantly reduces energy consumption as compared to uniform gossip (90%), spatial gossip (average 86%), and alge-gossip (aveage 60%). Also, NBgossip is identical to alge-gossip in terms of number of rounds to spread all the messages. What is more, we investigated how NBgossip can be applied in WSNs. By extending the original NBgossip to NBbroadcast, which makes use of the wireless broadcast advantage, the time and energy incurred is further reduced. Finally, we commented on the feasibility of implementation on existing sensor nodes like micaz and stargate.

In Chapter 4, we considered the problem of distributed averaging problem. This problem is of particular interest to many applications: data calibration, measurement

noise removal, distributed data fusion, and enroute data aggregation. By formulating the message selection process as a closest point search, the proposed CPSgossip protocol gets rid of the uncertainty involved in many existing algorithms. Under CPSgossip protocol, nodes are allowed to communicate with any other nodes in the network. Moreover, each node asks its gossip partner for the message it desired the most rather than blindly combining values with partner. Towards the end, the performance of CPSgossip in wireless sensor network is evaluated and compared with other approaches. The CPSgossip consumes the least amount of energy and always return the correct value without any probability involved.

We realized that the performance of sensors will gradually degrade over time. Therefore, sensor data calibration is required to ensure data integrity. Throughout Chapter 5, we reviewed calibration techniques for wireless sensor nodes and proposed a loopy belief propagation solution to solve the calibration problem. We elaborated how the experiment was set up and the potential functions are generated. What is more, we tested our approach on a real sensor network and proved the effectiveness of our proposed algorithm. The key idea is to exploit the spatial correlation among neighboring sensor nodes and apply loopy belief propagation to infer the MAP assignment of the real values. Last but not least, for data calibration, a promising research direction is mobility-enhanced calibration, where a mobile actuator could be used to calibrate static sensor nodes. By offering multiple sensing perspectives, mobility can detect calibration errors that may not be detected through coordination among statically collocated sensor nodes.

In this work, we propose a few novel algorithms to solve well-known existing problems. The algorithms have been proved mathematically and further confirmed through simulation and experiments. However, none of them have been tested out on real sensor nodes. Moreover, in this thesis, we have made a number of assumptions. For example, the number of sensor nodes is assumed to be known and constant during the data dissemination cycle. In reality, existing sensor nodes could die and new sensor nodes may join the network. Therefore, future extension can be developed to address the above mentioned needs.

Author's Publications

1. F Lu, LT Chia, KL Tay, WH Chong, "NBgossip: an Energy-efficient Gossip Algorithm for Wireless Sensor Networks", Special Issue on Selected Best Papers from IEEE MASS, Vol. 23, No.3, Journal of Computer Science and Technology, Springer, May, 2008.

2. Feng Lu, Liang-Tien Chia, "Closest Point Search for Computing Average", submitted to Elsevier Ad Hoc Networks Journal. (Extended version of SENSORCOMM paper, major revision)

3. Feng Lu, Lijuan Geng, Liang-Tien Chia, Ying-Chang Liang, "Secure Multi-path in Sensor Networks", in the 5th ACM Conference on Embedded Networked Sensor Systems (Sensys), Sydney, Australia, Nov, 2007.

4. Feng Lu, Liang-Tien Chia, Kok Leong Tay, "NBgossip - Neighborhood Gossip with Network Coding Based Message Aggregation", in the 4th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS), Pisa, Italy, Oct, 2007.

5. Feng Lu, Liang-Tien Chia, "Gossip-based Computation of Average: a Closest Point Search Approach", in the 1st IARIA/IEEE International Conference on Sensor Technologies and Applications (SENSORCOMM), Valencia, Spain, Oct, 2007.

References

[1] CM Christensen. "The innovator's delemma: when new technologies cause great firms to fail", Havard Business School Press, Boston, MA, 1997.

[2] G Moore. "Progress in digital integrated electronics", IEDM Tech. Digiest, pp. 11-13, 1975.

[3] S Borkar. "Design Challenges of Technology Scaling", IEEE Micro, Vol. 19, No. 4, pp. 23-29, 1999.

[4] Computer Science and Telecommunication Board. Embedded everywhere: a research agenda for networked systems of embedded computers, Washington, D.C.: National Research Council, 2001.

[5] JM Kahn, RH Katz, KSJ Pister. "Next century challenges: mobile networking for Smart Dust", in *Proc. ACM Conference on Mobile Computing and Networking (MobiCom)*, MA, USA, 2000.

[6] D Estrin, R Govindan, J Heidemann, S Kumar. "Next century challenges: scalable coordination in sensor networks", in *Proc. ACM Conference on Mobile Computing and Networking (MobiCom)*, Washington, USA, 1999.

[7] IF Akyildiz, W Su, Y Sankarasubramaniam, E Cayirci. "A survey on sensor networks ", IEEE Comm. Magazine, Vol 40, No.8, pp. 102-114, 2002.

[8] K Romer, F Mattern. "The design space of wireless sensor networks", IEEE Wireless Comm., Vol 11, No. 6, pp. 54-61, 2004.

[9] P Levis, D Culler. "Mate: a tiny virtual machine for sensor networks", ACM SIGOPS Operating Systems Review, Vol 36, No. 5, pp. 85-95, 2002.

[10] N Bulusu. "Self-configuration localization systems", PhD thesis, UCLA, 2002.

[11] G Hoblos, M Staroswiecki, A Aitouch. "Optimal design of fault tolerant sensor networks", in *Proc. IEEE International Conference on Control Applications*, 2000.

[12] A Demers, D Greene, C Houser, W Irish, J Larson. "Epidemic algorithms for replicated database maintenance", in *ACM Sympo. Principles of Distributed Computing*, Vancouver, Canada, Aug.1987.

[13] C Fragouli, J Widmer. "Network coding: an instant primer", *ACM SIGCOMM Computer Communication Review*, vol. 36, issue 1, Jan. 2006.

[14] The network coding homepage, http://www.networkcoding.info/, maintained by Ralf Koetter.

[15] T Ho, M Medard, J Shi, M Effros, DR Karger. "On randomized network coding", in *Proc.* 41^st *Allerton Annu. Conf. Communication, Control and Computing*, Monticelo, IL, Oct. 2003.

[16] J Dall, M Christensen. Physical Review E, 2002.

[17] DJ Watts, SH Strogatz. Nature (London) 393, 400, 1998.

[18] S Boyd, A Ghosh, B Prabhakar, D Shah. "Gossip algorithms: design, analysis and applications", in *Proc. of IEEE INFOCOM* Miami, UAS, 2005.

[19] D Snyder, M Miller. Random point processes in time and space , New York, 1991.

[20] D Kempe and J Kleinberg, and A Demers. "Spatial gossip an resource location protocols", in *Proc. ACM Symp. Theory of Computing*, Heraklion, Crete, Greece, Jul. 2001.

[21] R Karp, C Schindelhauer, S Shenker, B Vocking. "Randomized rumor spreading", in *Proc. Foundations of Computer Science*, Redondo Beach, CA, Nov. 2000.

[22] D Kempe, JON Kleinberg, A Demers. "Spatial gossip and resource location protocols", *Journal of the ACM*, Vol.51, No.6, pp.934-967, Nov. 2004.

[23] AG Dimakis, AD Sarwate, MJ Wainwright. "Geographic gossip: efficient aggregation for sensor networks", in *Information Processing on Sensor Networks* Nashville, Tennesses, USA, 2006.

[24] D Kempe, A Dobra, J Gehrke. "Gossip-based computation of aggregate information", in *Proc. of the 44th Annu. IEEE Symp. on Foundations of Computer Science*, Cambridge, MA, USA, 2003.

[25] S Deb, M Mdard, C Chout. "Algebraic gossip: a network coding approach to optimal multiple rumor mongering", *IEEE Trans. Information Theory*, vol. 52, no. 6, pp. 2486–2507, Jun. 2006.

[26] C Fernandess, D Malkhi. "On collaborative content distribution using multi-message gossip", in *Proc. of Parallel and Distributed Processing Sympo.*, Greece, Apr. 2006.

[27] R Ahlswede, N Cai, SYR Li, RW Yeung. "Network information flow", *IEEE Trans. Information Theory*, vol. 46, no. 4, pp. 1204-1216, Jul. 2000.

[28] Y Wu. "Network coding for multicasting", Ph.D. thesis, Princeton University, 2006.

[29] SYRY Li, RWN Cai. "Linear Network Coding", *IEEE Trans. Information Theory*, vol. 49, no. 2, pp. 371-381, Feb. 2003.

[30] T Ho, R Koetter, M Medard, DR Karger, M Effros. "The benefits of coding over routing in a randomized setting", in *Proc. IEEE Symp. Information Theory*, Yokohama, Japan, Jun. 2003.

[31] M Bawa, H Garcia-Molina, A Gionis, R Motwani. "Estimating aggregates on a peer-to-peer network", Technical report, Stanford University, 2003.

[32] S Boyd, A Ghosh, B Prabhakar, D Shah. "Analysis and optimization of randomized gossip algorithms", in *Proc. of 43rd IEEE Conference on Decision and Control*, Bahamas, Dec. 2004.

[33] P Gupta, PR Kumar. "The capacity of wireless networks", *IEEE Trans. on Information Theory*, Vol.46, No.2, pp.388-404, Mar. 2004.

[34] CC Moallemi, B Van Roy. "Consensus Propagation ", Technical report, Stanford University, Jun. 2005.

[35] D Mosk-Aoyama, D Shah. "Fast distributed algorithms for computing separable functions ", http://arxiv.org/abs/cs.NI/0504029

[36] E Agrell, T Eriksson, A Vardy, K Zeger. "Closest point search in lattices ", *IEEE Trans. on Information Theory*, vol. 48, No. 8, pp. 2201-2214, Aug. 2002.

[37] JH Conway, NJA Sloane. "Sphere Packings, Lattices and Groups ", ch.20, Springer-Verlag, 1999.

[38] S Khot. "Hardness of approximating the shortest vector problem in lattices ", *Journal of the ACM*, vol. 52, pp. 789 - 808, Sep. 2005.

[39] AE Gamal, J Mammen, B Prabhakar, D Shah. "Throughput-delay trade-off in wireless networks ", in *Proc. of IEEE INFOCOM*, Hong Kong, China, 2004.

[40] M Penrose. "Random Geometric Graphs ", Oxford University Press, Oxford, 2003.

[41] K Langendoen, N Reijers. "Distributed localization in wireless sensor networks: a quantitative comparison ", *Computer Networks*, vol. 43, pp. 499-518, Nov. 2003.

[42] IF Akyildiz, W Su, Y Sankarasubramaniam, E Cayirci. "Wireless sensor networks: a survey ", *Computer Networks*, vol. 38, pp. 393-422, Mar. 2002.

[43] T Melodia, D Pompili, IF Akyildiz. "Optimal local topology knowledge for energy efficient geographical routing in sensor networks", in *Proc. of IEEE INFOCOM*, Hong Kong, China, 2004.

[44] B Karp, HT Kung. "GPSR: greedy perimeter stateless routing for wireless networks", in *Proc. of ACM Conference on Mobile Computing and Networking*, Boston, USA, 2000.

[45] D Culler, D Estrin, M Srrvastava. "Overview of sensor networks", *IEEE Computer*, vol. 37, no.8, pp. 41-49, Aug. 2004.

[46] J Feng, G Qu, M Potkonjak. "Differential on-line sensor calibration", Sensors, 2004, *Proceeding of IEEE*, Oct, 2004.

[47] V Bychkovskiy, S Megerian, D Estrin, M Potkonjak. "A Collaborative Approach to In-Place Sensor Calibration", in *Information Processing in Sensor Networks*, USA, 2003.

[48] K Whitehouse, D Culler. "Calibration as parameter estimation in sensor networks", in *1st ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.

[49] AT Ihler, JW Fisher III, RL Moses, AS Willsky. " Nonparametric belief propagation for self-localization of sensor networks", *IEEE Journal on Selected Areas in Communications*, vol. 23, issue 4, pp. 809-819, 2005.

[50] V Cevher, JH McClellan. " Sensor array calibration via tracking with the extended Kalmanfilter", in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2001.

[51] S Ganeriwal, A Kansal, MB Srivastava. " Self aware actuation for fault repair in sensor networks", in *IEEE International Conference on Robotics and Automation*, 2004.

[52] JS Yedidia, WT Freeman and Y Weiss. "Understanding belief propagation and its generalizations", MERL TR-2001-22, Jan. 2002.

[53] Crossbow Technology Inc. at http://www.xbow.com